Predicting recurring concepts on data-streams by means of a meta-model and a fuzzy similarity function

Abad Miguel Ángel[a], Gomes João Bártolo[b], Menasalvas Ernestina[a,*]

[a] *Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, 28660 Boadilla del Monte, Madrid, Spain*
[b] *Institute for Infocomm Research (I2R), A*STAR, Singapore, 1 Fusionopolis Way Connexis #21, 138632, Singapore*

A B S T R A C T

Stream-mining approach is defined as a set of cutting-edge techniques designed to process streams of data in real time, in order to extract knowledge. In the particular case of classification, stream-mining has to adapt its behavior to the volatile underlying data distributions, what has been called concept drift. It is important to note that concept drift may lead to situations where predictive models become invalid and have therefore to be updated to represent the actual concepts that data poses. In this context, there is a specific type of concept drift, known as recurrent concept drift, where the concepts represented by data have already appeared in the past. In those cases the learning process could be saved or at least minimized by applying a previously trained model.

To deal with the aforementioned scenario, meta-models can be used in the process of enhancing the drift detection mechanisms used by data stream algorithms, by representing and predicting when the change will occur. There are some real-world situations where a concept reappears, as in the case of intrusion detection systems (IDS), where the same incidents or an adaptation of them usually reappear over time. In these environments the early prediction of drift by means of a better knowledge of past models can help to anticipate to the change, thus improving efficiency of the model regarding the training instances needed.

Furthermore, as a complement of meta-models, a mechanism to assess the similarity between classifica-tion models is also needed when dealing with recurrent concepts. In this context, when reusing a previously trained model a rough comparison between concepts is usually made, applying boolean logic. The intro-duction of fuzzy logic comparisons between models could lead to a better efficient reuse of previously seen concepts, by applying not just equal models, but also similar ones.

This work faces the aforementioned open issues by means of the MM-PRec system, that integrates a meta-model mechanism and a fuzzy similarity function. The theoretical proposal of MM-PRec is also validated in this paper by means of different experiments using both synthetic and real datasets.

1. Introduction

This work is framed within the scope of data stream classification (Gaber, Zaslavsky, & Krishnaswamy, 2007), which aims to learn a classification model incrementally from a stream of training records in order to use it later to predict the class of unlabeled records with high accuracy. Most of these kinds of classification models lack of an efficient adaptation to the environment where they are implemented which, in most cases, is constantly changing. For this reason, coping with the improvement and adaptation of classification algorithms on data streams is still a great challenge, as long as data stream mining imposes some requirements that have to be accomplished, namely: maintaining an efficient behavior in the system, i.e. stable computational and memory load; while providing suitable quality in the classification process, i.e. high accuracy of predictions.

Concept drift techniques have been extensively applied to cope with changes in the underlying distribution of records over time, allowing classification models to be able to adapt their behavior when needed (Gama, Medas, Castillo, Rodrigues, & Labidi, 2004; Tsymbal, 2004; Žliobaitė, 2010).

In cases where concepts reappear, a mechanism could be applied to allow them to be remembered, thus improving the performance of the learning algorithm. A concept drift means that the system is

* Corresponding author. Tel.: +34637569431.
E-mail addresses: miguel.abad.arranz@alumnos.upm.es (A. Miguel Ángel), bartologjp@i2r.a-star.edu.sg (G. João Bártolo), emenasalvas@fi.upm.es (M. Ernestina).

behaving in a different way from that expected, that is, a new kind of concept is probably taking place. But that is not always the case, as normal behavior can also change over time (Lane & Brodley, 1999). That is why an adaptive model should achieve a suitable accuracy in the classification process in case of a concept drift, adapting its behavior to a new situation when a malfunctioning of the system takes place. This means that the classification model should be adapted to provide a high level of accuracy by means of training on the new context. However there are situations in which a new training is not needed, as the new concept is equal or similar to a previous one. In these cases we could reuse a previously-trained model, saving computation costs and thus providing an efficient method to undertake this new context. But it is not just a question of reusing a previous model. It is also important to be able to detect in a proactive manner, and based on an evaluation of the behavior of the system, which is the most suitable model for a certain situation or context. If we had the possibility of training a meta-model representing the pattern of changes in the system, we would be able to predict the most similar model representing the current behavior, reaching an early detection of drift capability and saving computational costs in terms of training instances needed.

In real-world domains, most concepts are recurrent (Harries, Sammut, & Horn, 1998; Widmer, 1997). This means that a previously seen concept may reappear in the future and probably in a similar context (Harries et al., 1998; Widmer, 1997). However, only a few approaches explore it (Gama & Kosina, 2009, 2014; Gonçalves & Barros, 2013; Katakis, Tsoumakas, & Vlahavas, 2010; Widmer & Kubat, 1996; Yang, Wu, & Zhu, 2005). In these cases, concept changes are generally the result of context changes. If context information were available, it could be used to better understand recurring concept changes. However, only a small number of techniques explore context information when dealing with recurring concept changes (Gama & Kosina, 2014; Gomes, Sousa, & Menasalvas, 2010; Harries et al., 1998; Widmer, 1997).

Extending this idea, in this work we propose MM-PRec as a novel data-stream learning system to aid in the process of concept drift management while simultaneously learning a meta-model to predict when the drift will occur as well as the most suitable model to be reused if necessary.

MM-PRec implements a Hidden Markov Model (HMM) to train the meta-model. To deal with recurring concepts, we propose to send a set of instances representing the drift process to the HMM getting, as a result, the prediction of the most probable model to be reused from those stored in a repository. In case of a false alarm regarding a drift, the HMM would return the same model that is currently being used.

In addition, MM-PRec also implements a fuzzy logic function (Kosinski, Prokopowicz, & Slezak, 2005) to decide the level of similarity between models. This is a crucial aspect to improve the storing process in the repository effectively. When a model has to be stored in the repository, it is required to know if the concept that the model is representing is recurrent or not. In case of a recurrent concept, it should not be stored, as there are already previous models representing it. Moreover, this fuzzy function is used when HMM is not available, being useful also for recurrence detection.

This process helps the system as a whole to save memory consumption, as long as just the models needed are stored in the repository. In this way, the proposed mechanism makes it feasible to work with complex data-stream environments where an overloaded repository would make it difficult to achieve a suitable quality of the system.

Due to the versatility of wrapper methods, the approach we propose is based on a wrapper mechanism, avoiding to solve the problem just for a specific algorithm.

This work presents several contributions: (i) integration of a HMM based meta-model to aid in the prediction drift process; (ii) integration of contextual information both when detecting drift and when retrieving models; (iii) use of a fuzzy similarity function to compare concepts.

To the best of our knowledge, this is the first work to deal with concept drift by means of the use of meta-models based on Hidden Markov Models and fuzzy logic to predict similar previously seen concepts.

Experiments performed with different real and synthetic datasets show the reduction of training records needed by MM-PRec with no loose of accuracy in comparison with other approaches of the literature.

The rest of the paper is organized as follows. In Section 2, we summarize related work on concept drift and context-aware approaches in recurring concepts, the use of sequence classifiers, and concept similarity, which is followed in Section 3 by the preliminaries of the approach where the motivation, challenges and problem definition are stated. Furthermore, in Section 4, we propose MM-PRec as a solution to work in recurring concept drift environments, with a detailed description of its components and the algorithm used. Section 5 presents the results obtained by the experimentation phase. Finally Section 6 presents the main conclusions and discussion of future lines of research.

2. Related work

The approach that we propose in this paper relies on a sequence classifier meta-model to be trained while the base learner algorithm deals with recurring concepts from the incoming data stream. The trained meta-model has all the context information associated with previously learnt concepts in a way that will make it easy to retrieve a previously built model that represents a concept similar to the current one. A fuzzy logic function is used to achieve this goal.

Consequently, in this section we review and compare our proposal with methods that address the problems of: data stream classification; recurring concepts; context-aware approaches; sequence classification and conceptual equivalence.

An overview of the learning process under concept drift is presented in Žliobaitė (2010). Moreover a recent review of the literature related to the problem of concept drift is presented in Gama, Žliobaitė, Bifet, Pechenizkiy, and Bouchachia (2014), including a survey of adaptive learning processes. Existing strategies for handling concept drift and an overview of the most representative, distinct, and popular techniques and algorithms are also presented, covering the different facets of concept drift.

In particular, among the different challenges for adaptive learning systems published in Žliobaitė et al. (2012), our approach goes into two of them in depth:

- It facilitates the study of the model to understand its behavior in concept drift scenarios over time.
- It develops an adaptive tool, rather than an adaptive algorithm. This provides a better stability and robustness over time.

2.1. Data stream classification

There have been several techniques developed to achieve the challenge that arises when dealing with concept drift, be they algorithms adaptations or wrapper mechanisms (adaptive tools). New algorithms have appeared during the last years Gaber et al. (2007); Gama et al. (2004); Hulten, Spencer, and Domingos (2001); Street and Kim (2001); Tsymbal (2004); Žliobaitė (2010); Widmer and Kubat (1996), but some other related challenges have received far less attention. Such is the case of situations where the same concept or a similar one reappears, and a previous model could be reused to enhance the learning process in terms of accuracy and processing time

as in the case of Gama and Kosina (2009); Gomes et al. (2010); Katakis et al. (2010); Ramamurthy and Bhatnagar (2007); Yang et al. (2005); 2006).

In this way, most existing proposals do not exploit this aspect and have to learn new concepts from scratch even if they are recurrent as it is the case of Brzezinski and Stefanowski (2011) and Brzezinski and Stefanowski (2013) where an ensemble mechanism is used to deal with concept drift. Similarly, in Katakis et al. (2010) an ensemble is also used, but incremental clustering is performed to maintain information on historical concepts. In this way, the proposed framework captures batches of examples from the stream into conceptual vectors. Conceptual vectors are clustered incrementally according to their distance and for each cluster a new classifier is learnt. Classifiers in the ensemble are then learnt using the clusters. The work of Elwell and Polikar (2011) proposed Learn++.NSE, an extension of Muhlbaier, Topalis, and Polikar (2009) for non-stationary environments. Learn++.NSE is also an ensemble approach that learns from consecutive batches of data without making any assumptions on the nature or rate of drift. The classifiers are combined using dynamic weight majority and the major novelty is on the weighting function that uses the classifiers time-adjusted accuracy on current and past environments. To deal with resource constraints Hosseini, Ahmadi, and Beigy (2012) proposes a novel algorithm to manage a pool of classifiers when learning recurring concepts. More recently, Haque, Parker, Khan, and Thuraisingham (2014) presented a multi-tiered ensemble based method HSMiner to address the challenges that exist when labelling instances in an evolving Big Data Stream. The method is very costly as it requires building large number of AdaBoost ensembles for each of the numeric features after receiving each new data chunk. Thus, three approaches to build these large number of AdaBoost ensembles using MapReduce based parallelism are presented.

Furthermore, in Hewahi and Elbouhissi (2015) a new approach called Concepts Seeds Gathering and Dataset Updating algorithm CSG-DU is presented to deal with data stream classification. CSG-DU is concerned with discovering new concepts in data stream and aims to increase the classification accuracy using any classification model when changes occur in the underlying concepts. The paper presents experimentation on synthetic and real datasets showing that the classification accuracy increased from low values to high and acceptable ones.

Moreover, in Mena-Torres and Aguilar-Ruiz (2014) a new technique, named Similarity-based Data Stream Classifier (SimC)is introduced. This technique achieves good performance by introducing a novel insertion/removal policy that adapts quickly to the data tendency and maintains a representative, small set of examples and estimators that guarantees good classification rates. The methodology is also able to detect novel classes/labels, during the running phase, and to remove useless ones that do not add any value to the classification process. Statistical tests were used to evaluate the model performance, from two points of view: efficacy (classification rate) and efficiency (online response time). Five well-known techniques and sixteen data streams were compared, using the Friedman's test. Also, to find out which schemes were significantly different, the Nemenyi's, Holm's and Shaffer's tests were considered. The results show that SimC is very competitive in terms of (absolute and streaming) accuracy, and classification/updating time, in comparison to several of the most popular methods in the literature.

Finally, in Kosina and Gama (2015) the very fast decision rules (VFDR) algorithm is presented together with interesting extensions to the base version. As algorithms designed to work with data streams should be able to detect changes and quickly adapt the decision model, in the paper the adaptive extension (AVFDR) to detect changes in the process generating data and adapt the decision model is also presented. Detecting local drifts takes advantage of the modularity of the rule sets. In AVFDR, each individual rule monitors the evolution of performance metrics to detect concept drift. AVFDR prunes rules whenever a drift is signaled. The experimental evaluation shows that the presented algorithms achieve competitive results in comparison to alternative methods and the adaptive methods are able to learn fast and compact rule sets from evolving streams.

The main drawback of these methods, apart from the computational process time needed, is the need of constantly train the models used being them recurrent or not.

2.2. Recurring concepts

Regarding similar methods to the one presented in this paper, able to deal with concept recurrence, these are their main characteristics:

- In the approach proposed in Gomes et al. (2010) context-concept relationships are learnt from the concept history. A model from a previously learnt concept associated with a particular context is reused in situations of recurrence. Moreover, the proposed method does not require the partition of the dataset into small batches. The concept representations are learnt by a base learner algorithm from an arbitrary number of records. These concept boundaries are determined when a drift detection method signals a change/drift. To improve Gomes et al. (2010), which relies on a single classifier (Naive Bayes) to deal with recurring concepts, the use of ensembles has been proposed in Gomes, Menasalvas, and Sousa (2011). The main difference between this system and the one proposed in this paper is the similarity function, that in our case allows to better fit the equivalence between classification models. Moreover, the implementation of meta-models allows to come early to recurrent drift detection, improving the estimation of recurrence provided by a single classifier as it is the case of Naive Bayes. However, both systems are composed by a two level framework: a base learner where an incremental algorithm learns the underlying concept; and a detection drift layer where the relations context-concepts are learned.
- RCD (Gonçalves & Barros, 2013) is a recent recurring concept drift framework that uses a non-parametric multivariate statistical tests to check for recurrence. In the case of RCD, statistical comparisons are made in order to detect recurrence, so it is needed to store different models and the buffer of instances associated to them. While this is also needed in the system presented in this paper, the implementation of meta-models avoid the need to make statistical comparisons with all the previously seen stored models to detect recurrence, as it is the case of RCD. Furthermore, the use of meta-models allows to better represent the context associated to concepts, in contrast to what occurs with raw buffers of instances.
- In the work of Ramamurthy and Bhatnagar (2007) the authors present an ensemble approach that exploits concept recurrence, using a global set of classifiers learned from sequential data chunks. If no classifier in the ensemble performs better than the error threshold, a new classifier is learned and stored to represent the current concept. The classifiers with better performance on the most recent data form part of the ensemble for labeling new records. The main drawback of this system compared to the MM-PRec presented in this paper is the computational resources needed to execute the ensemble method. Furthermore, the efficiency of the ensemble method depends on the number of classification models used. In contrast, the meta-model presented in this paper, while posing some computational restrictions, allows to centralize the concept recurrence prediction by means of the Hidden Markov Model.
- A system that monitors the evolution of the learning process is presented in Gama and Kosina (2014). The system uses meta-learning techniques that characterize the domain of applicability of previously learned models. The meta-learner can detect

recurrence of contexts, using unlabeled examples, and take proactive actions by activating previously learned models. However, the main difference between this system and the one proposed in this paper is that MM-PRec needs only one meta-learner, while the former develops one meta-model attached to each model being learned. Furthermore, MM-PRec meta-learner is based on a multi-instance classifier, allowing to accurately represent the patterns of drifts and their context information, while the meta-models proposed in Gama and Kosina (2014) are based on single-instance classifiers.

- The method proposed by Yang et al. (2005) consists of using a proactive approach to recurring concepts, which means reusing a concept from the concept history. This concept history is represented as a Markov chain which allows the most probable concept to be selected according to a given transition matrix. This could be seen as simplification of a meta-model just representing the changes from one concept to another. However, the MM-PRec presented in this paper allows to generate a context-concept relationship in a way such that it is possible to predict not the next state of a Markov chain, but the most appropriate model to be used for a specific context using pattern recognition techniques. Furthermore, the concept history storage is also improved by MM-PRec thanks to the fuzzy similarity function, that avoids duplicate similar classification models.

In sum, in this paper we present the use of a meta-model based on Hidden Markov models to predict future similar behaviors regarding concept drift. Hidden Markov models have been proved to accurately deal with pattern recognition, and therefore in this paper they are used to represent and detect the patterns associated to contexts. Furthermore, the fuzzy similarity function improves any other similarity function based on crisp logic, allowing also to go deep into the variables that characterize different classification models. Having mentioned the main differences between the MM-PRec system presented here and other similar methods, it is important to note that the main drawback of MM-PRec is the training process of the meta-model, that must be done in a batch mode. As a consequence, the preprocessing of the context information and the training process of the Hidden Markov Model may delay in some cases the stream mining learning process.

2.3. Context-aware approaches

Context dependence has been recognized as a problem in several real world domains (Harries et al., 1998; Turney, 1993; Widmer, 1997). Turney (1993) was among the first to introduce the problem of context in machine learning, where he presented a formal definition in which the notions of primary, contextual and context-sensitive features were introduced. Such notions are based on a probability distribution for the observed classes given the features.

Widmer (1997) exploits what is referred to as contextual clues (based on the Turney definition of primary/contextual features (Turney, 1993)) and proposes a meta-learning method to identify these clues. Contextual clues are context-defining attributes or combinations of attributes whose values are characteristic of the underlying concept. When more or less systematic changes in their values are observed this might indicate a change in the target concept. The method automatically detects contextual clues on-line, and when a potential context change is signaled, knowledge of the recognized context clues is used to adapt the learning process in some appropriate way. However, if the *hidden context* is not represented in the contextual clues, that is, if the reason behind the change is not represented in the feature space, it is not possible to detect or adapt to the change.

In Zliobaite et al. (2012) the authors aims to identify the key research directions to be taken to bring the adaptive learning closer to application needs identifying six challenges: making adaptive systems scalable, dealing with realistic data, improving usability and trust, integrating expert knowledge, taking into account various application needs, and moving from adaptive algorithms towards adaptive tools.

The conceptual clustering approach proposed by Harries et al. (1998), identifies stable *hidden contexts* from a training set by clustering the instances assuming that similarity of context is reflected by the degree to which instances are well classified by the same concept. A set of models is constructed based on the identified clusters. This idea has proven to work well with recurring concepts and real world problems. However, its main drawback is the off-line training required to obtain the conceptual clusters, as these could lead to inaccuracy with concepts or patterns that were not seen during training.

More recently in Žliobaitė, Bifet, Read, Pfahringer, and Holmes (2015) the authors theoretically analyze evaluation of classifiers on streaming data with temporal dependence suggesting that the commonly accepted data stream classification measures, such as classification accuracy and Kappa statistic, fail to diagnose cases of poor performance when temporal dependence is present. Therefore they should not be used as sole performance indicators. The authors develop a new evaluation methodology for data stream classification that takes temporal dependence into account proposing a combined measure for classification performance, that takes into account temporal dependence.

Finally, in Forkan, Khalil, Tari, Foufou, and Bouras (2015) pattern recognition models for detecting behavioral and health-related changes in a patient who is monitored continuously in an assisted living environment is described. The early anticipation of anomalies can improve the rate of disease prevention. In the paper a Hidden Markov Model based approach for detecting abnormalities in daily activities, a process of identifying irregularity in routine behaviors from statistical histories and an exponential smoothing technique to predict future changes in various vital signs is presented. The outcomes of these different models are then fused using a fuzzy rule-based model for making the final guess and sending an accurate context-aware alert to the health-care service providers. In the paper the authors also evaluate some case studies for different patient scenarios in ambient assisted living. Although this work is similar to this paper in the sense that it implements Hidden Markov Models and fuzzy logic, the main difference is that in this work these components are used to detect drifts as an external layer. In contrast, the work of Forkan et al. (2015) use them as a base learner, detecting abnormal behaviors for a specific context as it is the case of health problems and therefore it is not suitable to deal with concept recurrence.

2.4. Sequence classification

There are several works available dealing with the classification of sequences (Xing, Pei, & Keogh, 2010). Among these, some of them use hidden Markov models (Bicego, Murino, & Figueiredo, 2004; Blasiak & Rangwala, 2011; Rabiner, 1989). These models are widely used in pattern recognition environments related to handwriting (Hu, Brown, & Turin, 1996), gestures (Eickeler, Kosmala, & Rigoll, 1998) or image recognition (Ghoshal, Ircing, & Khudanpur, 2005). However, to the best of our knowledge, there are no mechanisms that use this kind of systems to deal with concept drift comprehension.

2.5. Conceptual equivalence

To determine whether a certain model represents a new concept or a reappearing one, a similarity measure is required. The current work is an improvement of the *Conceptual equivalence* measure proposed by Yang, Wu, and Zhu (2006) where a fuzzy logic function (Mendel, 1995) is used to better represent the relationship between different concepts.

3. Problem definition and preliminaries

This section provides the necessary background to understand MM-PRec system. We start by motivating and defining the problem, explaining some real world situations where a system like the one proposed in this paper could be useful, as well as some basic definitions to understand the basics of the solution proposed. All the challenges that we have had to face during the development and implementation of MM-PRec are outlined.

From now on we assume that the data streams that are used as input in the MM-PRec model are already preprocessed and adapted to work well with incremental data stream classification processes. In this way, we can then assume that we do not need to preprocess the data streams, this work being out of the scope of this paper.

3.1. Motivation

Data stream mining algorithms must come up with the problem of having to keep in memory just a limited number of records to train their models. That is why these algorithms have to cope with the task of processing each training record only once, while maintaining a suitable quality on the resulting model. This is the main difference from traditional data mining algorithms, where multiple passes over data are common.

In particular, that leads to classification techniques on data streams where models have to be learned incrementally with each incoming record. With the availability of these kinds of models, it is feasible to predict the class of unlabeled records anytime from an early stage. Obviously, the more training records the better precision values obtained.

However, in scenarios where the data distribution changes the accuracy of the classification is expected to decrease. In these cases, to continuously maintain the quality of the models, it is also important for them to be able to detect and adapt anytime to changes in the underlying concept that they represent (Tsymbal, 2004).

The aforementioned changes in the underlying concept can be caused by different factors. In the present work we focus in the following ones:

- Context changes, either hidden or explicit (Gama et al., 2004; Harries et al., 1998; Tsymbal, 2004; Widmer & Kubat, 1996) that lead to concept drift.
- Recurring concepts, as a particular type of the aforementioned concept drift (Gama & Kosina, 2009; Katakis et al., 2010; Widmer & Kubat, 1996; Yang et al., 2005) where a previous learned concept is expected to reappear.

We envisage that recognizing and predicting already learned concepts might help the system to better adapt to future changes where these concepts reappear. With that recognition task in place, it would be possible for the algorithms to avoid relearning something from scratch that has been already learned (Gama & Kosina, 2009, 2014; Gomes et al., 2010; Gonçalves & Barros, 2013; Katakis et al., 2010; Widmer & Kubat, 1996; Yang et al., 2005). This same idea has been already explored in Gomes et al. (2010), where concepts are able to be saved in a repository. Actually, MM-PRec can be seen as an extension of the MRec system presented in Gomes et al. (2010).

However in our approach we propose a fuzzy based mechanism to decide about similarity of models, improving the storage of the models. On the other hand, our approach also differs from that presented in Gomes et al. (2010) as in our case we add a meta learner that is able to predict when recurring concepts will occur. Therefore we can anticipate to recurrent drifts choosing also the most appropriate model for the incoming context. As a result, the number of instances needed for the training process is expected to decrease when recurring concepts appear.

3.2. Real world cases

3.2.1. Intrusion detection system

An intrusion detection system (IDS) is a typical monitoring problem which aims to detect cyber incidents. In this case, a trained classification model could send alerts to the operator when a malfunction in the system occurs. But to use such a classification model for an IDS effectively, we must ensure that the IDS is able to adapt to concept drift. A concept drift in an IDS means that the system is behaving in a different way from that expected. But that different behavior may be caused by a new kind of intrusion that is probably taking place, or because the system monitored is changing in a controlled environment (no intrusion is taking place).

In any case, the IDS should adapt its classification model to the new situation. If we were able to store all the patterns that represent the different situations of the system monitored (its concepts), we could reuse previously seen models easily. In this way, imagine the case of a central models manager to which each individual IDS connects to in order to check if a specific concept is recurrent or not. In a system like this, the central component would be responsible for executing and training the meta-model. The meta-model would be trained based on the information sent by the different IDS. Therefore, local IDS would be responsible for sending the different patterns associated to a specific concept or situation. As a result, the local IDS would benefit from the knowledge hold by the meta-model, saving training instances when dealing with recurrent drifts, and improving their behavior in a collaborative way.

3.2.2. Fraud detection

A similar situation like the one explained in the case of IDS, would be the case of a set of systems dealing with fraud detection. Taking into account that each different fraud detection system should be able to deal with concept drift, the context associated to each concept managed by them could be sent to a central mechanism. This central mechanism could use the information provided by the different local systems to develop a meta-model able to detect similar and recurrent concepts. In a system like this, the individual fraud detection systems would benefit from the experience of the rest of systems when dealing with drift along time.

3.3. Preliminaries

3.3.1. Learning with concept drift

Let X be the space of attributes with its possible values; Y the set of possible discrete class values. Let D be the data stream of training records arriving sequentially $X_i = (\vec{x_i}, y_i)$ with $x_i \in X$ (feature space) and $y_i \in Y$, where $\vec{x_i}$ is a vector of the attribute values and y_i is the (discrete) class label for the i^{th} record in the stream. In order to train a base learner based on a classification model m incrementally, these records are processed by m with the goal of predicting the class label of a new record $\vec{x} \in X$, so that $m(\vec{x}) = y \in Y$.

As stated in Yang et al. (2006), the concept term is more subjective than objective. That is why in the scope of this paper a concept is represented by the learning results of the classification algorithm used as a base learner, such a Hoeffding Tree (Domingos & Hulten, 2000).

In this field, we consider that a stable concept has been learned when the records used during a given period k are independently and identically distributed according to a probability distribution $P_k(x, y)$. In these situations where concept change, $P_k(x, y) \neq P_{k+1}(x, y)$.

We have to take into account that a change of concept can be abrupt or gradual. Not all the solutions presented to deal with change of concepts are suitable both to abrupt and gradual changes. In gradual concept changes is where we can get the most out of MM-PRec. This is because the meta-model of the MM-PRec, as will be detailed later, has to be trained with the records involved at the time in which concept drift takes place. Therefore in case of an abrupt change of

concept, the meta-model will not have enough records to be trained and the goodness of the MM-PRec will not be as appropriate as in gradual changes.

3.3.2. Recurring concepts

A recurring concept change can be detected when the input records during a period k are generated based on the same distribution as a previously observed period, in a way that $P_k(x, y) = P_{k-j}(x, y)$. To deal with these kinds of situation, the model m_k learned from a certain period k could be saved to be reused later if it is needed. This would avoid the need to learn a new model representing the same concept as m_k. With this solution the continuous learning process improves its behavior, not requiring a previously learned concept to be learnt from scratch. In addition this approach needs fewer training records to be processed than other approaches that do not deal with recurrent concepts. However, to better calculate whether a concept is recurrent or not, a similarity function is usually used. This is the case of the similarity function proposed in Yang et al. (2006), which is the starting point in developing the new fuzzy similarity method proposed in this paper.

3.3.3. Hidden Markov models

Hidden Markov Models (HMM) are known to work extremely well in practice as prediction, recognition, and identification systems, etc., in a very efficient manner. These kinds of system are developed under the assumption that the process they have to represent can be well characterized as a parametric random process. Taking into account that HMM are a good resource in dealing with pattern recognition, here they are used to recognize the patterns that arise from each concept drift that appears in a data stream classification model.

A Markov process is one for which its output is related to a set of states at each instant of time, where each state corresponds to a physical and observable element. HMM extend that case to include situations where the observation is a probabilistic function of the state.

In order to implement HMM in a specific scenario, we have to decide what the states in the model are, and how many of them should be in the model. Therefore there are multiple HMM models to solve a specific problem, but practical considerations impose some strict limitations on the size of models that we can consider. An HMM is made up of the following (Rabiner, 1989):

- N, as the number of states in the model. Although these states are "hidden", there is often some physical significance attached to the set of states of a model. It is common to allow states to interconnect in such a way that any state can be reached from any other state, which it is commonly known as "egordic" models. Individual states are denoted as $S = \{S_1, S_2, \ldots, S_N\}$, and the state at time t as q_t.
- M, the number of distinct observation symbols per each state, i.e., the discrete distinct concepts learned. These symbols correspond to the physical output of the system being modeled, being denoted as $V = \{v_1, v_2, \ldots, v_M\}$.
- The state transition probability distribution $A = \{a_ij\}$ where $a_ij = P[q_{t+1} = S_j \mid q_t = S_i]$, $1 \leq i, j \leq N$.
- The probability distribution for each observation symbol in state j, $B = \{b_j(k)\}$, where $b_j(k) = P[v_k \text{ at t} \mid q_t = S_j]$, $1 \leq j \leq N$ and $1 \leq k \leq M$.
- The initial state distribution $\Pi = P[q_1 = S_i]$, $1 \leq i, j \leq N$.

To reach a complete specification of a HMM, we should provide the following parameters: number of states of the model (N); number of output values (M); specification of observation values used as input; and specification of the three probability measures (A, B and Π).

With a specific HMM being defined, three main challenges must be addressed if we want the model to be useful in real-world applications. These main challenges are:

1. The evaluation problem, referring the computation of the probability of an input sequence observations by a HMM. The probability here refers to the likelihood of an observation sequence of being produced by the HMM, which is also known as the test phase of a classifier model.
2. The problem of finding the optimal-state sequences for a specific input observation sequence. This problem deals with the challenge of uncovering the hidden part of the HMM.
3. The problem of optimizing the model parameters to best describe how a given observation sequence comes about. This problem is related to the training process of a classifier model.

One of the main contributions of this paper is focused on providing an effective approach for training and testing a HMM while assuming the limitations that this model poses, as it has been proven that HMM models provide excellent results in real world scenarios. The main aforementioned limitations are referred to:

- HMM models are based on the assumption that consecutive observations are independent, and therefore the probability of a sequence of observations $P(O_1, O_2, \ldots, O_T)$ can be written as the product of probabilities of individual observations in a way such $P(O_1, O_2, \ldots, O_T) = \prod_{i=1}^{T} P(O_i)$.
- Another limitation comes from the Markov assumption itself. This assumption refers to the fact that the probability of being in a given state at time t only depends on the state at time $t - 1$, not taking into account dependencies between several states.

In the scope of this work, the use of HMM has resulted in the improvement of concept-drift detection and prediction.

Further below we explain the rest of the challenges related to detecting and predicting recurrent concept drift on data-stream environments.

3.4. Challenges

The main challenges we deal with in this paper are:

- Learning a meta-model trained with information related to the drifts that have occurred in the history of the system. The meta-model will be able to predict similar drifts in the future.
- Arranging a fuzzy similarity function in order to better calculate the level of similarity between different concepts.
- Detecting concept drift as soon as possible.
- Adapting as soon as possible to recurrent concept drift by using the aforementioned meta-model.
- Setting up a tool to better understand concept drifts by means of the meta-model.

MM-PRec approach faces these challenges by training a HMM for the implementation of the meta-model and a fuzzy function to deal with concept similarities evaluation.

3.5. MM-PRec

MM-PRec is made up of two main elements depicted in Fig. 1:

- A meta-model based on a sequence classification algorithm using Hidden Markov Models (HMM). This component allows MM-PRec system to predict both when the drift will happen and the most suitable concept for each situation if it is recurrent. In order to do so, a repository of previous models is used.
- A concept similarity function based on fuzzy logic. This function is used to determine the level of similarity between concepts. This fuzzy similarity function is crucial to solve the problem of deciding not just which is the most suitable model, but also if the storage of a specific model in the repository is required.

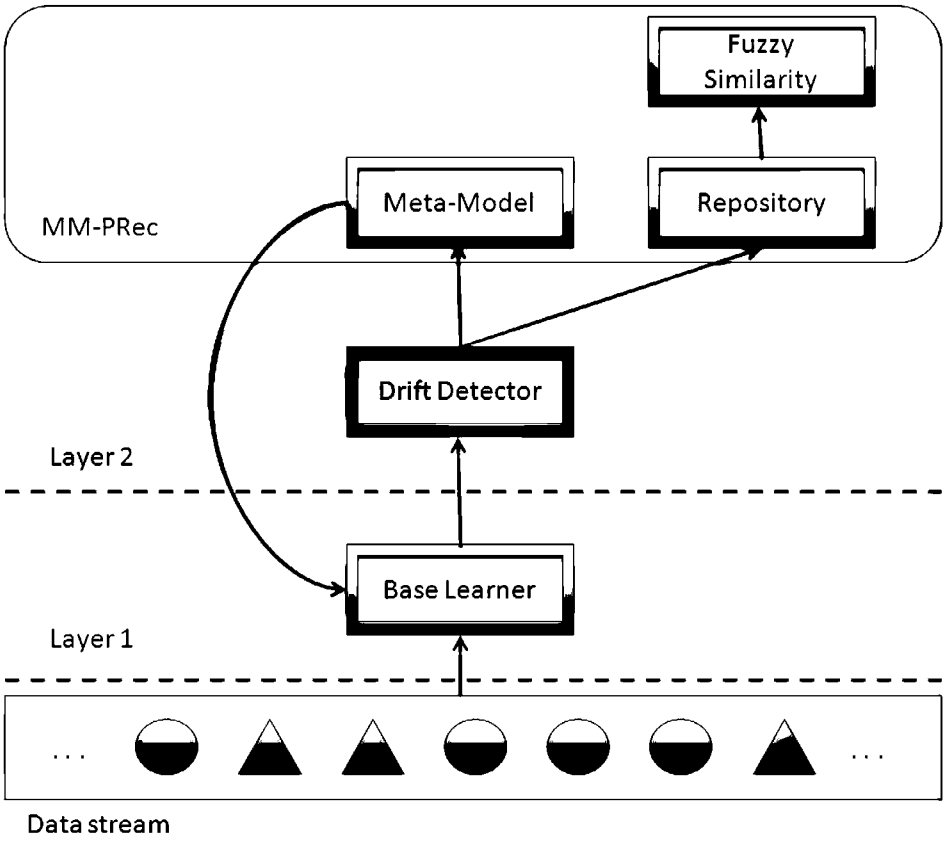

Fig. 1. MM-PRec Components.

This is therefore a substantial improvement in the work presented in Gomes et al. (2010), where the problem of concept drift in recurring scenarios was solved in a similar way also by using a repository of concepts and a crisp similarity function, but with the lack of a meta-model to explain and predict concept drifts at an early stage.

The proposed MM-PRec system allows us to better deal with recurrent situations in a classification problem in data streams, helping the evolving base learner to adapt to drifts. This is achieved by predicting when the drift will happen from a bunch of records at a particular time; and also by getting a similarity level between concepts from an improved fuzzy similarity function. All these new features make MM-PRec an effective system to obtain the most suitable model for a given context. Hence the MM-PRec system is a feasible tool to be used in a wide range of real application scenarios.

3.5.1. Concept similarity

We propose a concept similarity function that uses fuzzy logic and it is based on that presented in Yang et al. (2006). This function is defined by the following parameters:

- A conceptual degree of equivalence based on the matching of two different models classifying many instances, even when their classifications are both wrong. It is not therefore an accuracy equivalence, but a measure of the level of both models to classify in the same way.
- A measure that represents the difference in the number of records used to train each model. This parameter is intended to provide a measure on the maturity and stability of the model.

From the aforementioned two parameters, a fuzzy (Zadeh, 1965) similarity value is estimated from a previously defined set of rules, making it possible to obtain the poor, average or high values.

The meta-model proposed is able to predict recurrent drifts as well as the most appropriate model to be used. However there are two situations where a similarity function is required in a complementary way:

1. A model must be stored in the repository of previously trained concepts: in this case the fuzzy similarity function is used to assess the need to store a new model. If there is a similar model in the repository, storing a similar one would not improve the quality of the classification process while unnecessarily increasing the memory consumption.
2. A drift is taking place, there is no meta-model available, and it is time to decide whether the new concept is recurrent or not: in this case, taking into account that the meta-model is not ready, the system has been training two different models in parallel to adapt to the drift in a recurrent way (the current model and a new one).

3.5.2. Meta model

Let S be the space of attributes representing sequential records (appearing during concept drift) with their possible values; Z the set of possible discrete class values representing the new concept to which the model has been adapted (therefore there is not a direct relationship between Z and Y, the later representing the class values of the original dataset). S being the space of sequential record attributes, this space is made up of several values of the feature space $X_i = (\vec{x_i})$ with $x_i \in X$ presented in Section 4.2.2.

Let W be the window of sequential records involved in the process of concept drift $S_i = (\vec{s_i}, z_i)$ with $s_i \in S$ (feature space) and $z_i \in Z$, where $\vec{s_i}$ is a vector of attribute values and z_i is the (discrete) class label (representing the new concept associated to the sequential records) for the i^{th} record in the stream. With this information a meta-model can be trained each time a concept drift is detected. In this way, a classification (meta) model p is trained by processing the incoming sequential records $\vec{s} \in S$. Once the meta-model p has been

trained, it could be possible to predict the class label (new concept to be used) from the new record used as input, such as $p(\vec{s}) = z \in Z$.

The benefits of having a meta-model arise from the utility of using the training records that appear during concept drift to learn what is going on in this drift. With this goal in mind, two main benefits are provided by the proposed meta-model:

- First of all, the meta-model can be used to predict similar concept drifts to those previously learnt.
- The meta-model can also be used to better understand the process of each concept drift, through the study of its internal behavior and representation through a meta learner algorithm. This would allow a "white box" instead of a "black box" to be used.

In this context, the records needed to train the proposed meta-model are not typical independent instances but a bunch of instances that represent a new concept itself (the concept drift). Therefore to train the meta-model it is better to use a mechanism that takes this issue into account. This is the case in the aforementioned Hidden Markov Models, that are able to learn sequential data (Bicego et al., 2004; Dietterich, 2002) in order to predict similar situations in the future based on the information provided during the training phase. This could produce an increment on the evaluation time needed by the model because of this additional training.

4. Implementation of the solution proposed

As in the case of the MRec system proposed in Gomes et al. (2010), MM-PRec can be seen as a two-layer framework:

1. A basic layer where an incremental learning algorithm is able to represent the underlying concept by means of a classification model.
2. An extended layer in which detection and adaptation to concept changes takes place. The detection of recurrent concepts is implemented in this layer. It is also at this level where MM-PRec implements its meta-model and fuzzy similarity mechanisms.

However, as can be seen from Fig. 1, MM-PRec presents two main improvements from the system proposed in Gomes et al. (2010), namely:

1. Implementation of a meta-model to represent, detect and predict recurrent drifts, by means of a HMM.
2. Implementation of a fuzzy similarity method to represent equivalences between different models.

To provide an in depth knowledge of the implementation of the MM-PRec system proposed in this paper, first the learning process is described in Section 4.1; the implementation of the meta-model learning process is described in Section 4.2; finally, the description of the fuzzy similarity concepts is presented in section 4.3.

4.1. The learning process

This section presents the way in which MM-PRec is integrated in the learning process of a data mining system. As a consequence, it shows how all the aforementioned elements that compose MM-PRec are used during the learning process.

The continuous learning process with the intervention of MM-PRec comprises the following steps:

1. The base learner processes the incoming records from data streams by means of an incremental learning algorithm to generate a decision model *currentClassifier* representing the underlying concept. Therefore this model will be used to classify unlabelled records.
2. A repository *MR* is created to allow concept representation storage.
3. A drift detection method *DriftDetection* is continuously monitoring the error-rate of the learning algorithm Gama et al. (2004). When the error-rate goes beyond some predefined levels, the drift detection method signals a *warning* (possible drift) or a *drift*. In case of a *warning* signal, a *newLearner* is trained to deal with the new coming concept, and a *WarningWindow* is activated to store the context information. However, as it can be seen in Section 4.2.2, any other drift detection method might be used.
4. At the same time a meta-model is trained from the context information provided by the drift detection method stored in the aforementioned warning window. In this way, and taking into account that the meta-model is a batch learner, the multi-instance training dataset must be adapted to each training phase, and the meta-model has to evolve at the same time as new concept drifts appear.
5. Throughout the life cycle of the system, three different cases may be used to adapt to changes in the underlying concept, depending on the availability or not of a trained meta-model:
   (a) The concept similarity method detects that the underlying concept is new, and the base learner has to learn it by processing the current incoming labelled records in an incremental way.
   (b) The fuzzy concept similarity method detects that the underlying concept is recurrent, and a previous model is applied.
   (c) The meta-model is able to predict the recurrent concept, determining the best model to be used from the repository *MR*.

It is important to note that the main advantage of reusing previously seen models is that they are no longer trained as they are stable models that represent adequately specific concepts. Therefore, in those cases where models are reused, the learning process is speed up and the number of needed training instances decrease.

The details of the on-line learning process for the proposed global learning system, as well as the method to detect and adapt to recurrent concepts are detailed in Algorithm 1. Let X be the set of stream instances that the learning system handles. In this context, specific records in the form $X_i = \{\vec{x}, y\}$ with $\vec{x} \in X$ are processed as they come, being $\vec{x}$ the set of attributes of the instance and y the class value associated to it.

During the stream processing of the learning system, different steps are accomplished to detect and manage recurrent drift. This behavior can be summarized as follows, referring to specific lines of the algorithm 1:

- In line 4, the drift detection method identifies the suitable drift level (*stable, warning or drift*).
- If the process is at the normal level (line 7), the base learner represented by the *currentClassifier* is updated with the new training record. This is the same behavior as in any other traditional data mining model ready to work with data streams.
- In the case of a warning level (line 8), if the repository does not have the *currentClassifier*, or a similar one as referred to in Section 4.3, the *currentClassifier* is stored in the repository *MR*. The storage process and the similarity check are implemented by means of the fuzzy similarity function. In addition (line 13) if there are enough records (it may vary for each problem to be solved) to send to the meta-model as multi-instance data, this data is sent to it in order to predict a recurrent concept as well as the best model to use from the repository as detailed in Section 4.2. If the meta-model returns a recurrent model to be used (line 14), this model is set as the *currentClassifier*, the drift detection method is restored to start with the information provided by this new model, and the meta-model is trained with the current meta-data. Still at this warning level (lines 19, 20 and 21), a *newLearner* is updated with the training record; the training record is also added to a *warningWindow*; and the dataset used to train the meta-model (meta-data) is updated with the information provided by the

Algorithm 1 Data Stream Learning Process.

Require: Data stream *DS*, ModelRepository *MR*

```
1: repeat
2:    Get next record X_i from DS;
3:    prediction = currentClassifier.classify(X_i);
4:    DriftDetection.update(prediction);
5:    switch DriftDetection.level
6:    case Normal
7:    currentClassifier.train(X_i);
8:    case Warning
9:    if ¬MR.containsSimilar(currentClassifier) then
10:       MR.store(currentClassifier);
11:   end if
12:   if history(c_o) > ρ then
13:      predictedModel = meta-model.getPrediction(history(c_o));
14:     if ¬predictedModel.isEmpty() then
15:        currentClassifier = predictedModel;
16:        meta-model.Update();
17:     end if
18:   end if
19:   WarningWindow.add(X_i);
20:   newLearner.train(X_i);
21:   meta-model.addInstances(X_i,newLearner.ID);
22:   case Drift
23:   repeat
24:      WarningWindow.add(X_i);
25:      newLearner.train(X_i);
26:   until WarningWindow.size > τ //Stability Period
27:   if ¬MR.containsSimilar(newLearner) then
28:      currentClassifier = newLearner;
29:   else
30:      currentClassifier = MR.getEquivalent(newLearner);
31:   end if
32:   meta-model.addInstances(X_i,currentClassifier.ID);
33:   meta-model.Update();
34:   case FalseAlarm
35:     WarningWindow.clear();
36:     newLearner.delete();
37:   end switch
38: until END OF STREAM
```

current *warningWindow* and the ID of the *newLearner* as the class of the meta-model. The *warningWindow* contains the latest records (which should belong to the most recent concept), and will also be used to calculate the conceptual equivalence and estimate the accuracy of models stored with the current concept.

- When drift is signalled (line 22), the *newLearner* is trained until a stability period is reached. This stability period is a variable that defines the number of instances that must be processed by the *warningWindow* during the drift level to make the *newLearner* suitable to deal with the new concept. When the stability period is over (line 26) it is compared with models stored in the repository *MR*. These comparisons are made in terms of conceptual equivalence as stated in Section 4.3, specifically by means of the fuzzy similarity function. If the underlying new concept is recurrent, a stored model is reached from the repository. This stored model is therefore used to represent the recurring underlying concept. In case there are not any equivalent model in the repository, the *newLearner* is finally used to deal with the new concept. It is important to remark that the benefit of implementing a previously seen model is that it does not need to be trained again, as it is supposed to be a stable model. When using the newLearner, it needs to be constantly trained during the learning process as it is still an immature model. Therefore, if newLearner is used there is not a decrease in the number of training instances needed. However, the risk of reusing a not suitable recurrent model is still latent. In those cases, the accuracy of the classification base learner would drop. Also at this stage the *warningWindow* is added to the dataset used to train the meta-model in the form of a bag of multi-instance data linked to the ID of the new learner used (i.e. the *newLearner* or that restored from the repository). Note that the algorithm will use this drift signal just in the case there are no meta-model available, or that the meta-model has not predicted any suitable model for the current underlying concept.
- A false alarm (line 34) case is used when a warning is signaled but then the learner returns back to normal without achieving drift. In those cases, both the *warningWindow* and the *newLearner* are cleared.

4.2. The metamodel approach

In order to learn a meta-model from the context information provided each time a concept drift takes place, a HMM is used. The reason for using a HMM is that these kinds of model have proved to be an excellent mechanism to deal with pattern recognition. In this work we therefore assume that a concept drift can be seen as a bunch of records representing a pattern from which it may be possible to predict similar ones. HMM are used here then as a sequential classification mechanism trained from the records involved in a concept drift.

As was stated in Section 3.3.3, to let HMM be useful we must carry out three main tasks, namely: a training phase of the HMM from sequential data; a testing phase where the system returns the probability of a new sequence having been generated from the HMM, and therefore it can also be used as a prediction measure; and the establishment of the hidden part of the system, which refers to the different states and the possible probability connections between them. This work is focussed on the training and prediction capacities of HMM, so below you will find all the details of the implementation used in the MM-PRec system.

4.2.1. HMM training

The meta-model implementation of MM-PRec needs to manage context-concepts associations not just to train the (meta) classifier, but also to predict recurrent drifts. Taking into account that in the case of MM-PRec the context information used is composed by the sequence of training instances used during the time in which the drift is taking place, MM-PRec needs to implement a mechanism to deal with the representation of the association that exist between a set of instances (bag) and a class value. This scenario is based in the "collective assumption", that states that the class label of a bag is a property that is related to all the instances within that bag. As a consequence, a multi-instance classification algorithm must be trained to allow the meta-model being effective. In the case of the work presented in this paper, the training process of the meta-model is achieved as follows:

1. Each time a concept drift is detected by the drift detector (no matter what detection mechanism is used), an additional base learner *newLearner* is trained to deal with the new concept. Therefore, during the time in which the drift is taking place, two parallel base learners are being trained (one to deal with the "old" concept, and the other to deal with the new appearing concept).
2. Once the *newLearner* fits to the new concept, improving the precision results provided by the original base learner, we can state that drift has taken place and *newLearner* becomes the unique base learner of the system. In this step, the warning window (set of instances that are being used when drift is appearing) associated to the drift is attached to the identification number (ID) of the *newLearner* as the class. This data is ready to be used as a single record in HMM and therefore is added to the dataset used as the training set for the meta-model.

3. If the MM-PRec requires the meta-model to be trained, the training dataset of the meta-model is used to accomplish that task. A minimum number of instances to train the meta-model must be set, taking into account that an insufficient number of training records can lead to unstable prediction models Rabiner (1989).

During the training process of the meta-model, some issues may arise:

- In those cases where there is just one record attached to a specific ID, the meta-model will not be able to be trained appropriately. This scenario would lead to the training of an over fitted classification model for the affected class value. Therefore, in those cases the meta-model is not able to predict similar contexts to the one associated to the class value, as long as there is just one record to represent it. In sum, this scenario would lead to a misunderstanding of the meta-model behavior, and as a result the HMM returns an error when this situation appears.
- In some cases the ID attached to the records refers to a model that has finally not been stored in the repository. Two different scenarios might be the cause of this issue:
  - Scenarios in which there is not a stable model during drift. As it has been already mentioned, when drift is detected by the drift detector, a parallel *newLearner* is created. During this process, noise may appear, which leads to a situation where that *newLearner* is replaced by different *newLearner* during time, until stability in the precision results is reached. In those cases, different warning windows representing the new appearing context may be associated to the first *newLearner* that was trained. However, when drift ends, we realize that the *newLearner* that must be linked to all those previous warning windows (that represent the context information) is the last one.
  - Scenarios in which similar models to the *newLearner* being created are already stored in the repository. Once the bags of instances representing context are linked to an ID class value that identifies a classification model, by means of the similarity function we can state that such ID refers to a model equivalent to another stored in the repository. In those cases, the classification model stored in the repository prevails, so its model ID is the one that must be used, and not the one associated to the *newLearner*.

In order to deal with the aforementioned problems regarding the training phase of a meta-model, a pre-processing of the training meta-model dataset has to be developed. In this case, the preprocessed dataset is the result of applying two stages:

1. To deal first with the cases where IDs refer to non-existent models in the repository, two different solutions are implemented depending on the origin of the problem:
   (a) In those cases where the problem is due to the nonexistence of a stable model during drift, the affected IDs are changed with the value of the next model used in a stable way. This is done to deal with those situations in which a concept drift passes through the warning phase several times before being effective. In those situations, some multi-instance records related to the same concept drift may refer to different model IDs that were temporary. That is why we need to adapt the class value to the last stable model used.
   (b) If the cause is due to the existence of similar models in the repository, the affected IDs are changed with the ID value of the equivalent model stored in the repository. As it has been mentioned, as long as the model stored in the repository overcomes others being similar, the IDs must be adapted in that way.
2. Besides, once the adaptation phase has taken place, we have to deal with the problem of "isolated" IDs, referring to those records where the ID used as class appears in them just once. In these cases, assuming that the ID they refer to are correct (the associated model exists in the repository), we have no other option than to erase them. However, they are erased temporarily just for the current training process. If there is a new training and new records attached to previously "isolated" IDs appear, they will take part in the training process.

During the meta-model training phase, the multi-instance algorithm is used as a batch classification learner that can be updated dynamically any time a drift is detected. However, a continuous re-training of the meta-model may lead to an overloaded system. Being aware of that, the approach presented in this work includes the definition of a MM-PRec parameter representing the minimum number of instances needed before training. Therefore, this parameter sets the number of instances that have to be processed by MM-PRec before committing a new training of the meta-model. This parameter allows to adapt MM-PRec suitably to the different real-problems in which recurrence could be used, fitting the training of the meta-model to the scenario where it is being used.

4.2.2. Drift detection mechanism

The MM-PRec system needs to know when a concept drift is taking place from the behavior of a base learner. For this purpose MM-PRec uses the method proposed by Gama et al. (2004). This method is based on the constant observation of the precision values of the base learner, calculating the error-rate of the learning process. This method is also based on a forgetting mechanism in which when drift appears, a new model is created to represent the new appearing concept.

Furthermore, as the most interesting feature of this method, it distinguishes three different stages or "drift levels". From those different drift levels we can determine the best moment when the meta-model could be asked to predict drift, taking into account that the context information associated to drift must be sent. In particular, the warning level refers to the moment when the error-rate starts to rise. That is the moment when the warning window starts to be filled with instances that could be sent to the meta-model in order to predict recurrent drifts. Besides, the out of control level is used to store in the repository the new learner created to deal with the drift, in those cases where the meta-model has not predicted recurrence.

In short, the following characteristics of this method are used in MM-PRec:

- The system assumes the observation of periods of stable concepts followed by changes that lead to new stable periods with different underlying concepts.
- The error-rate of the base learning algorithm is considered as a random variable from a sequence of Bernoulli trials.
- The general form of the probability of detecting an error is given by means of a binomial distribution.
- Three different drift levels are defined to manage concept changes: stable or at a control level, warning level and drift or out of control level. These levels represent the confidence of the mechanism of having detected a concept drift.

However, as the main focus of MM-PRec is dealing with recurrent concept drifts, this method has been extended to provide the required connection with the other elements that compose MM-PRec system. The solution presented in this paper is based on that presented in Gomes et al. (2010) in which a similarity function is used to assess whether the coming concept is recurrent or not. The solution presented in MM-PRec extends that function using fuzzy logic to improve the similarity detection and meta-models to predict drifts, overcoming the problems that the solution of Gomes et al. (2010) possess.

It is also important to note that other similar drift detection methods can be used in MM-Prec. Since the MM-PRec system has been developed as a wrapper mechanism, the specific method used for it is transparent, so it is not necessary to change the learning process.

4.3. Concept similarity

There are two situations in which MM-PRec has to decide if a concept is recurrent or not:

- When a model representing a concept is going to be stored in the repository. In this case, if the concept has appeared before, the model should not be stored as it would lead to duplicate models.
- When a new learner is trained to deal with the drift. In this case, MM-PRec must check with the models in the repository if the concept is recurrent or not.

In both cases, a similarity function is required to achieve the purpose of comparing different models determine an equivalence degree. An innovative feature of this research is the proposal of a fuzzy logic system Cox (1992) to calculate the *Conceptual equivalence* measure between classification models.

The term "fuzzy logic" was introduced in Zadeh (1965), and is a way of representing many-valued logic, allowing approximate reasoning to be applied through the definition of variables with several truth ranges (from 0 to 1) and rule sets. A rule set determines which fuzzy operator must be used in each case.

By means of using fuzzy logic, it is easy to deal with the concept of partial truth, where a truth value may range from completely true to completely false. In fuzzy logic applications it is common to use linguistic variables to facilitate the implementation of rules and truth values. In this way, a linguistic variable may have several truth values in the same system. These truth values can be seen as subranges of a continuous variable.

In the proposed MM-PRec system, three linguistic variables are defined:

- The variable *equal_classified*, used to represent the similarity in the classification behavior of two different models, may take the values: poor, good and excellent.
- The variable *diff_training*, used to represent the difference that exist in the number of training records used between two different models, may take the values: small and big.
- The variable *similarity*, a variable use to calculate the output of the fuzzy system based on the aforementioned variables, may take the values: poor, average and high.

The variable *equal_classified* is based on the method proposed by Yang et al. (2006) to calculate its conceptual equivalence. In our case, as it has been outlined, the equivalence between two models when dealing with classification similarity is treated as one parameter of the global fuzzy function. This parameter is calculated as follows:

1. Given two classification models m_1, m_2 and a sample dataset D_n of n records, it is possible to calculate for each instance $X_i=(\vec{x_i}, y_i)$ a score, $\text{score}(D_n) = +1$ if $(\text{prediction}(m_1(\vec{x_i})) = \text{prediction}(m_2(\vec{x_i})))$
2. $\text{score}(D_n)$ is used to represent the degree of equivalence in the classification process between m_1 and m_2.
3. The final classification equivalence *ce* value, that is a continuous value score with range [0,1], is calculated by

$$ce = \frac{score(D_n)}{N}$$

Depending on the value of *ce*, *equal_classified* will take one or another membership value, as represented in Fig. 2, where we can see

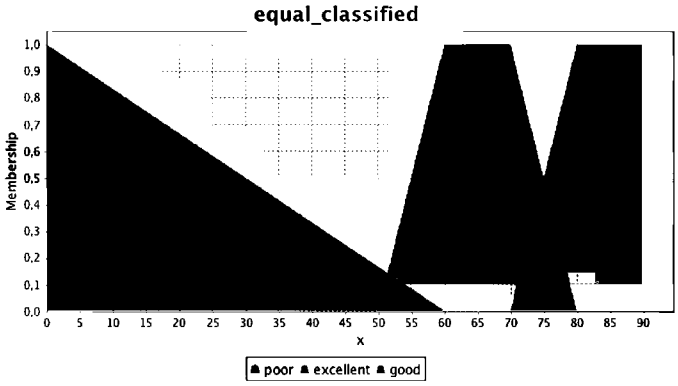

Fig. 2. Membership function of variable equal_classified.

the values this variable may take. The larger the output value of *ce*, the higher the degree of classification equivalence. For the records in D_n it compares how m_1 and m_2 classify the records. As in Yang et al. (2006), the similarity in the classification processes is not necessarily related to the accuracy attribute. This means that two models that present low accuracy for a set of records will have a high *ce* value, and therefore a high *equal_classified* value.

As regards the variable *diff_training*, its value represents the difference in the number of instances used to train each model we are trying to compare. In Fig. 3 we can see the membership values this variable may take.

The defuzzification method "Center Of Gravity" presented in Cox (1992) is used to calculate the final value of the *similarity* variable representing the conceptual equivalence, it being a very popular method in which the "center of mass" of the result provides the crisp value. The rule set is defined as follows:

1. IF *equal_classified* IS *poor* OR *diff_training* IS *big* THEN *similarity* IS *poor*.
2. IF *equal_classified* IS *good* AND *diff_training* IS *big* THEN *similarity* IS *poor*.
3. IF *equal_classified* IS *good* AND *diff_training* IS *small* THEN *similarity* IS *average*.
4. IF *equal_classified* IS *excellent* AND *diff_training* is *big* THEN *similarity* is *average*.
5. IF *equal_classified* IS *excellent* AND *diff_training* is *small* THEN *similarity* is *high*.

From the crisp value returned by the defuzzyfication method, we evaluate if it is above a predefined threshold. In that case, we assume that the models are similar and thus represent the same underlying concept.

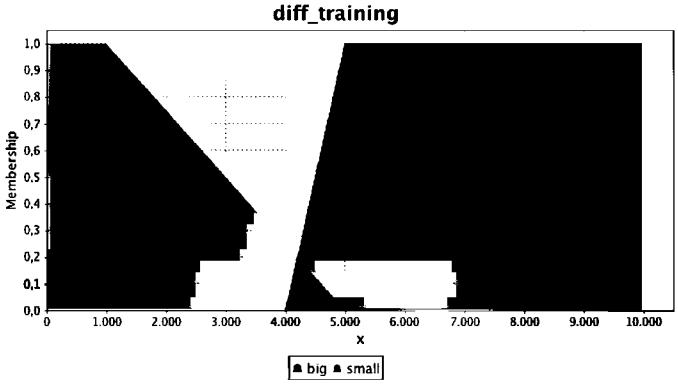

Fig. 3. Membership function of variable diff_training.

5. Experiments

In order to validate the MM-PRec method, and taking into account that MM-PRec is an extension of the MRec method cited in Gomes et al. (2010), three different experiments have been developed:

1. Experiment 1: The goal of this experiment is to prove that the precision of MM-PRec is similar to the MRec method, and no worse than other methods able to deal with concept drift. In order to do so, accuracy and kappa statistic measures are evaluated.
2. Experiment 2: The goal of this experiment is to prove that the training instances needed by MM-PRec when drifts appear are fewer than the ones needed when using MRec. Also an increment on the evaluation time needed is expected when using MM-PRec because of the meta-model training.
3. Experiment 3: The goal of this experiment is to prove that MM-PRec provides better precision results than an Active Classifier using the same percentage of instances.

In addition a statistical analysis has been developed to validate the results provided by the execution of experiments 1, 2 and 3.

To sum up, the main goal of this experimentation phase is to test the feasibility of using a fuzzy similarity procedure and a meta-model to predict similar previously seen concept drifts, testing its suitability when dealing with different kind of drifts.

5.1. Parameters setting

To develop the aforementioned experiments, both synthetic and real datasets have been used. A description of the different datasets applied is presented below along with the specific similarity threshold values set in each case for the MRec and MM-PRec methods. This similarity threshold must be established to afford the comparison process between models, so different values are set depending on the type of the dataset used. This is important because we must assure that the reused models really fit the context of the data during the learning process. Hence, lower values of the similarity threshold would lead to reuse models that may be not appropriate to the new concept in course. In contrast, higher values would make MRec and MM-PRec to look for previously seen models that really fit the concept represented by data. In situations where noise could be present in the data, it is important to set higher similarity threshold values to avoid misconceptions.

Furthermore, as stated in Section 3.3.3, to reach a complete specification of a HMM we should provide the following parameters: number of states of the model (N); number of output values (M); specification of observation values used as input; and specification of the three probability measures (A, B and Π). In all the experiments, N=5 and M will vary in each training phase depending on the number of models that are stored in the repository. As regards the probability measures, they are randomly initialized.

Below a description of the different datasets used during the experimentation phase is made.

5.2. Datasets

5.2.1. SEA dataset

This synthetic dataset is made up of 5M of records as a result of the merge process of five different SEA (Street & Kim, 2001) datasets, each of them containing 1M of records. Besides, each of the 5 SEA datasets are characterized by extended periods without drift with occasional concept changes, and the last one is characterized by a higher level of noise. In particular, each individual dataset was created varying the function parameter of the SEA generator. As long as there are four different values, the first dataset was created with a function value of 1, the second with a function value of 2, the third with a function value of 3, the fourth with a function value of 4 and the fifth with a function value of 1 again. In this last case, the noisePercentage parameter was set to 30 while in the other datasets the noisePercentage was established to its default value of 10.

The merge process of such five original SEA datasets results in a unique dataset characterized by abrupt drifts.

In order to better assess the behavior of each algorithm when dealing with this merged dataset, the analysis of the experiments made with it is separated into five different data chunks. The similarity threshold used is set to 0.8 both in MM-PRec and MRec methods. Taking into account the occasional soft drifts and the noise in the data, this value guarantees a high precision rate while reusing similar models.

5.2.2. Hyperplane dataset

A different synthetic dataset with gradual drifting concepts was created based on a moving hyperplane. A hyperplane in d-dimensional space is denoted by equation: $\sum_{i=1}^{d} a_i x_i = a_0$. Instances are labeled as positive if $\sum_{i=1}^{d} a_i x_i \geq a_0$, and as negative if $\sum_{i=1}^{d} a_i x_i < a_0$. Hyperplanes have been used to simulate time-changing concepts because the orientation and the position of the hyperplane can be changed in a smooth manner by changing the magnitude of the weights (Hulten et al., 2001). This dataset contains 170,000 instances and it represent different recurrent drifts. In particular, it is composed of 9 different concepts of 10,000 instances each where 8 of them are recurrent (they appear 2 times). The parameters used for the Hyperplane generator in MOA were: 2 classes, 10 attributes, 2 attributes with drift, a noise percentage of 10 and the sigma percentage varying from 10 to 50. More specifically, the first 10,000 instances were created with a sigma value of 10, the second with a value of 15, the third with a value of 20 and so on.

Taking into account that some noise has been introduced to the dataset, a threshold value of 0.9 is set to ensure that the reused models really fit the concept represented by the data when drifts happen.

5.2.3. Airlines dataset

This real dataset was first used for classification purposes in Žliobaitė, Bifet, Pfahringer, and Holmes (2011), and contains 539,384 records. It represents whether a flight was delayed or not from some information about it, i.e. the airline, the airports involved, or the day of week.

As long as it is difficult to establish what the actual concept drifts are, a generic assessment is made on the entire dataset. In this way, taking into account that we have no evidences of recurring concept drifts, we set a similarity threshold value of 0.7 to let the implementation of previously trained models that represented concepts similar to the existing one every moment. A higher value would restrict the use of previously seen models.

5.2.4. Electricity dataset

The Electricity Market Dataset (Elec2) (Harries, 1999) is a real dataset that uses data collected from the Australian New South Wales Electricity Market, where the electricity prices are not stationary and are affected by the market supply and demand. The market demand is influenced by context such as season, weather, time of the day and central business district population density. In addition, the supply is influenced primarily by the number of on-line generators, whereas an influencing factor for the price evolution of the electricity market is time. During the time period described in the dataset, the electricity market was expanded with the inclusion of adjacent areas (Victoria state), which led to more elaborate management of the supply as oversupply in one area could be sold interstate.

The Elec2 dataset contains 45,312 records obtained from 7th May 1996 to 5th December 1998, with one record for each half hour (i.e., there are 48 instances for each time period of one day). The class label identifies the change in the price related to a moving average of the last 24 hours. As shown in Harries (1999), the dataset exhibits

substantial seasonality and is influenced by changes in context. Taking into account that this dataset is expected to have gradual or soft drifts, a similarity threshold of 0.9 is used for this dataset in order to force both MRec and MM-PRec to reuse just the models associated to concepts really similar to the new appearing one in case of a drift.

5.2.5. Poker dataset

Poker-Hand dataset is a real set of 829,201 instances composed by 11 attributes. Each record of the dataset is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes. There is one class attribute that describes the "Poker Hand". The dataset used is the one normalized available on the MOA (Holmes, Kirkby, & Pfahringer, 2007) webpage.

Being difficult to establish what the actual concept drifts are and taking into account that we have no evidences of recurring concept drifts, we set a similarity threshold value of 0.7 to let a more flexible implementation of previously trained models.

5.2.6. Sensor dataset

Sensor stream (Zhu, 2010) is a real dataset that contains information (temperature, humidity, light, and sensor voltage) collected from 54 sensors deployed in Intel Berkeley Research Lab. The whole stream contains consecutive information recorded over a 2 months period (1 reading per 1–3 min) which makes a total of 2,219,803 instances. The learning task of the stream is to correctly identify which of the 54 sensors is associated to the sensor information read. The goal of this experiment is to effectively detect and adapt to the multiple concept drifts that this dataset contains.

Taking into account that recurrent drifts are expected to appear in this dataset, a similarity threshold of 0.9 is set in order to force both MRec and MM-PRec to use previously seen models just in case there were a high level of certainty of equivalence between concepts.

5.3. Environment

The implementation of the MM-PRec learning system was developed in Java, using the MOA (Holmes et al., 2007) environment as a test-bed. The specific components implemented in MM-PRec were developed using jFuzzyLogic (Cingolani & Alcala-Fernandez, 2012) for the fuzzy similarity function and Weka-HMM (Gillies, 2010) for the meta-model development.

During the execution of the different experiments, the following MOA evaluation features were established:

1. The *Prequential-error* method (Holmes et al., 2007) as the main evaluation technique.
2. The *HoeffdingTree* (Domingos & Hulten, 2000) class as base learner.
3. The *SingleClassifierDrift* class as the method in charge of detecting drifts. This class implements the drift detection method of Gama et al. (2004) and adapts to drift by learning a new classifier (i.e., discards previous concept representations).

It is important to note that just one system has been used to implement both the base learner and the meta-model in MM-PRec method, so no distributed environment has been available for the execution of the experimentation phase.

In order to develop the statistical analysis R (Team, 2010) software was used with the "coin" and "multcomp" packages. Taking into account that when comparing several methods over multiple datasets a post-hoc analysis is usually desired, in this case the post-hoc tests would be developed if needed using the Wilcoxon–Nemenyi–McDonald–Thompson test (Hollander & Wolfe, 1999), using the code of (Galili, 2010).

Finally, it is important to note that during the precision results the Kappa statistic (Cohen, 1960) values are included. Cohen's kappa coefficient is a statistic which measures inter-rater agreement for items. It is generally thought to be a more robust measure than simple percent agreement calculation, since takes into account the agreement occurring by chance.

As it is stated in Bifet and Frank (2010), accuracy is only appropriate when all classes are balanced, and have (approximately) the same number of examples. In order to cover the rest of cases, the authors propose the Kappa statistic as a more sensitive measure for quantifying the predictive performance of streaming classifiers. Just like accuracy, Kappa needs to be estimated using some sampling procedure. Standard estimation procedures for small datasets, such as cross-validation, do not apply. In the case of very large datasets or data streams, there are two basic evaluation procedures: holdout evaluation and prequential evaluation (the one used in the experiments of this paper). Only the latter provides a picture of performance over time. In prequential evaluation (also known as interleaved test-then-train evaluation), each example in a data stream is used for testing before it is used for training. In sum, the authors argue that prequential accuracy is not well-suited for data streams with unbalanced data, and that a prequential estimate of Kappa should be used instead. For that reason, and in order to better assess precision values, we have included both values (accuracy and kappa) when dealing with precision analysis.

5.4. Results

A description of the results obtained during the execution of the different experiments presented in the beginning of Section 5 is made below. All the experiments have been executed on the datasets presented in Section 5.2.

5.4.1. Experiment 1

The goal of this experiment was to prove that the precision values (accuracy and kappa statistic) provided by MM-PRec were similar to ones provided by the MRec method, and no worse than other different methods able to deal with concept drift.

Specifically, apart from the MM-PRec, the methods used to evaluate the results of accuracy and kappa statistic provided by the execution of this experiment have been:

1. The Hoeffding Tree Adaptive method (HT-ADWIN) presented in Bifet and Gavaldà (2009).
2. The AUE ensemble method (AUE method has been proved to deal well with recurrent concept drifts, although it does not provide a reduction of the training instances needed) presented in Brzezinski and Stefanowski (2013), using 10 Hoeffding Tree classifiers on it.
3. RCD method presented in Gonçalves and Barros (2013) using HoeffdingTree classifier.
4. MRec presented in Gomes et al. (2010) with HoeffdingTree class as base learner.

When applying this experiment to the SEA dataset, we can see from the measures represented in Table 1 that the AUE ensemble method provides the better results on both accuracy and kappa. Looking closely at the behavior of MM-PRec on the different chunks that shape this dataset we can state that:

1. During the first chunk the behavior of the three algorithms measured is quite similar. The reason why MM-PRec behaves in this way may be due to the fact that it has not yet detected recurrent drifts during the learning process.
2. During the second chunk MRec provides a lower precision rate, while MM-PRec behaves still well. Although AUE is the most precise method, the values provided by MM-PRec during this chunk denote that this method is not too far from it. Thanks to the fuzzy similarity function implemented, the MM-PRec method is able to deal with the abrupt drift presented in the beginning of this chunk, improving the precision values provided by MRec.

Table 1
SEA dataset precision.

Chunk		HT-ADWIN		AUE		RCD		MRec		MM-PRec	
		Acc.	Kappa	Acc.	Kappa	Acc.	Kappa	Acc.	Kappa	Acc.	Kappa
1M	Mean	89.46	76.51	**89.69**	**77.01**	88.15	73.25	89.41	76.39	89.41	76.39
	Std.Dev.	0.87	1.97	1.23	2.43	0.95	2.17	0.95	2.15	**0.95**	**2.15**
2M	Mean	89.46	78.31	**89.72**	**78.83**	88.11	75.29	42.97	0.63	89.28	77.92
	Std.Dev.	**0.93**	**1.93**	**0.93**	1.97	1.16	2.43	4.23	6.3	1.12	2.36
3M	Mean	89.54	73.56	**89.76**	**74.11**	83.36	61.33	89.04	72.83	81.7	58.2
	Std.Dev.	**0.93**	**2.33**	1.06	2.47	2.31	4.5	5.06	6.55	1.17	2.61
4M	Mean	89.43	78.68	**89.69**	**79.21**	79.43	57.72	79.6	58.08	81.49	62.1
	Std.Dev.	**0.97**	**1.97**	1.18	2.45	1.56	3.1	1.3	2.57	1.1	2.18
5M	Mean	79.59	59.17	**79.86**	**59.72**	78.62	57.23	79.75	59.49	78.92	57.84
	Std.Dev.	1.47	2.93	1.44	2.88	**1.31**	**2.62**	1.35	2.69	1.36	2.73
TOTAL	Mean	87.49	73.25	**87.74**	**73.78**	83.53	64.98	76.15	53.48	84.16	66.49
	Std.Dev	**4.09**	**7.61**	4.11	7.66	4.36	8.37	17.41	27.75	4.49	9.17

Table 2
SEA dataset performance.

Chunk	HT-ADWIN		AUE		RCD		MRec		MM-PRec	
	Time(s.)	Training Inst.	Time(s.)	Training Inst.	Time(s.)	Training Inst.	Time(s.)	Training Inst.		
1M	58.67	1M	810.95	1M	33.16	1M	52.9	1M	60.42	1M
2M	60.11	1M	1034.05	1M	91.38	994,003	61.42	**9000**	136.57	996,995
3M	58.77	1M	968.88	1M	245.45	1M	**92.15**	984000	140.15	**3,988**
4M	62.89	1M	972.74	1M	428.72	1M	**64.35**	11000	108.8	**0**
5M	61.2	1M	868.21	1M	509.77	1M	**63.96**	**0**	2,229.87	**0**
TOTAL	301.65	100%	4,654.83	100%	1,308.48	99.88%	334.78	40.08%	2,675.81	**40.02%**

Table 3
Datasets precision.

Dataset		HT-ADWIN		AUE		RCD		MRec		MM-PRec	
		Acc.	Kappa	Acc.	Kappa	Acc.	Kappa	Acc.	Kappa	Acc.	Kappa
Airlines	Mean	63.63	18.19	**66.66**	22.16	65.08	**22.26**	64.65	20.49	64.85	20.92
	Std.Dev.	4.82	7.22	**4.69**	9.1	5.39	**6.54**	5.74	7.9	5.36	9.01
Elec2	Mean	83.88	63.02	77.54	52.27	79.2	56.22	84.74	67.94	**85.2**	**69.05**
	Std.Dev.	5.18	10.8	6.69	14.74	7.18	15.43	4.15	9.45	**3.54**	**7.31**
Poker	Mean	63.62	25.54	67.27	27.03	**76.07**	**47.07**	73.23	40.21	74.44	40.64
	Std.Dev.	10.49	16.35	11.73	18.58	10.52	17.85	9.65	17.18	**6.61**	**16.65**
Sensor	Mean	61.2	61.19	82.50	82.01	52.51	51.32	82.88	82.41	**85.72**	**85.31**
	Std.Dev.	30.18	30.98	16.17	16.67	19.57	20.1	18.2	18.67	**13.76**	**14.21**
Hyperplane	Mean	84.69	69.35	84.28	68.53	68.46	36.91	89.63	79.24	**90.09**	**80.17**
	Std.Dev.	5.28	10.57	7.22	14.46	10.97	21.92	1.96	3.91	**1.35**	**2.69**

3. During the third chunk the precision values of MM-PRec drop. In this case, the similarity function has reused a previously seen model that does not fit completely the actual concept.
4. During the fourth chunk the precision values of MM-PRec remain low. While dealing well with the abrupt drift in the beginning of this chunk, the MM-PRec method cannot recover the precision rate provided during the first and second chunks because it is again using a previously seen model. However, in this case the model used fits well with the actual concept.
5. During the fifth chunk of data characterized by a high level of noise, the precision values of all the methods assessed drop but being similar.

A similar behavior of MM-PRec occurs when using the Airlines dataset, as can be seen in Table 3. Although both MRec and MM-PRec have a slightly smaller accuracy and kappa statistic values than the AUE algorithm, MM-PRec provides higher precision values than MRec.

For the rest of the datasets represented in Table 3 we can see that by using MM-PRec method the higher values on accuracy and kappa are obtained, while keeping a low level of dispersion on them. That means that MM-PRec is the more precise method evaluated and also the one that provides a smoother behavior when using these datasets, specially when dealing with gradual datasets.

To sum up, we can conclude that MM-PRec provides similar values not just to MRec but also to the rest of methods compared. Furthermore, we can state that in all the datasets used, MM-PRec never obtains fewer precision values than MRec. When comparing the precision results of RCD, it is important to note that just in two cases MM-PRec provides lower rate than RCD (Airlines and Poker datasets). However in both cases the difference is not to large and it is caused by the presence of abrupt drifts, where RCD behaves better than MM-PRec.

5.4.2. Experiment 2

The goal of this experiment is to prove that the training instances needed by MM-PRec when drifts appear are fewer than the ones needed when using MRec and RCD. Also an assessment of the evaluation time of MM-PRec is made.

In the case of the SEA dataset, we can see in Table 2 that MM-PRec uses in total less training instances than MRec and RCD. Regarding MM-PRec, the bigger amount of instances needed while training are

Table 4
Datasets performance.

	HT-ADWIN		AUE		RCD		MRec		MM-PRec	
Dataset	Time(s.)	Training Inst.(%)	Time(s.)	Training Inst.(%)	Time(s.)	Training Inst.(%)	Time(s.)	Training Inst.(%)	Time(s.)	Training Inst.(%)
Airlines	**146.98**	100	2560.75	100	1,304.57	**55.17**	613.8	93.1	2005.94	93.1
Elec2	**1.14**	100	9.55	100	40.6	100	9.7	88.11	264.88	**87.72**
Poker	**13.97**	100	105.55	100	133.89	83.06	785.79	51.4	12,194.9	**4.12**
Sensor	**195.01**	100	1,750.64	100	1,037.11	100	4,038.19	89.95	15,697.42	**85.06**
Hyperplane	**3.35**	100	26.3	100	14.63	100	6.15	61.01	5,682	**59.86**

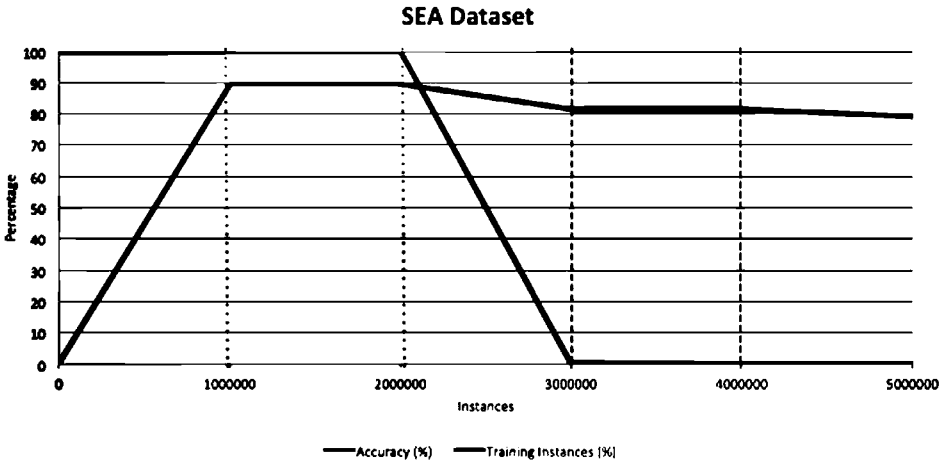

Fig. 4. SEA accuracy vs training instances.

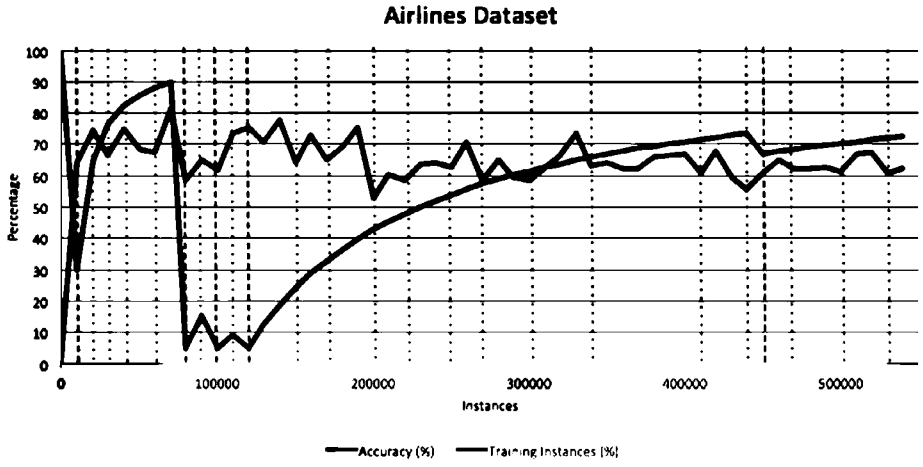

Fig. 5. Airlines accuracy vs training instances.

related to the first 2 chunks of data, which means that the prediction process during that time was not fitted yet to predict previously seen concepts.

Regarding the rest of the datasets presented in Table 4, we can see that in all cases MM-PRec uses fewer instances than MRec, except in the case of the Airlines dataset where MRec and MM-PRec use the same percentage and where RCD use fewer amount of training instances.

In Figs. 4, 5, 6, 7, 8, and 9 we can see a comparative of the accuracy of MM-PRec and the training instances needed during the learning process. Concept drifts are signaled by vertical lines, where the red dotted ones represent drifts where a new model is learned and the black dotted ones represent recurrent models (no training is needed).

When comparing the evaluation time presented in Table 2 for the SEA dataset and in Tables 4 and 5 for the rest of datasets, we can state that MM-PRec needs extra computational time in all cases. Furthermore, MM-PRec is the method that takes longer when evaluating the different datasets, except in the SEA and Airlines datasets, where AUE algorithm consumes more evaluation time than it. The cause of this increment in the time needed by MM-PRec is directly associated to the training time of the HMM meta-model. As long as the rest of algorithms assessed do not need any extra cost on processing streams, MM-PRec needs to keep the meta-model up to date. In fact MM-PRec makes a preprocessing of data in order to adapt the instances to the multi-instance environment needed in Weka-HMM which also increases the time needed by this method.

The conclusion of this experiment is that MM-PRec is the method that makes the most efficient use of training instances. However, it needs extra computational time when comparing it with the rest of the methods assessed, because of the preprocessing and training time needed by the meta-model.

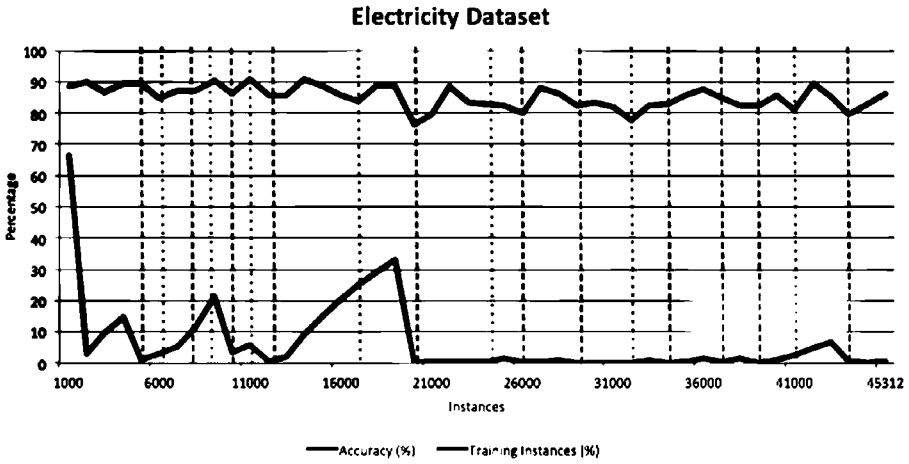

Fig. 6. Electricity accuracy vs training instances.

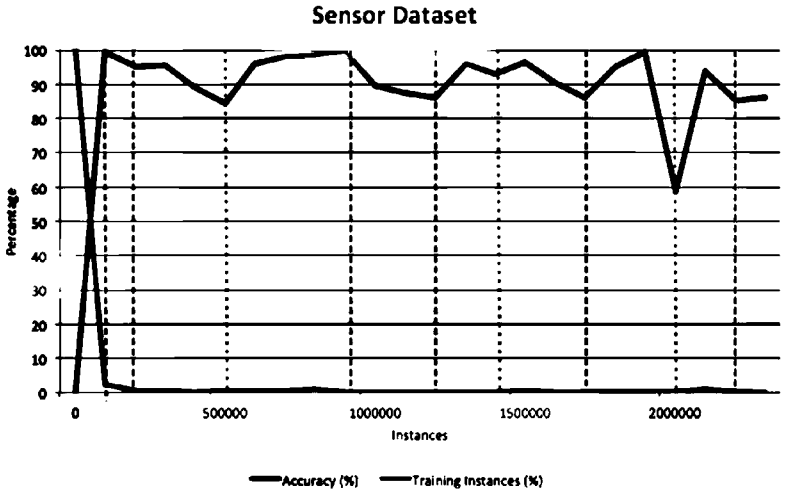

Fig. 7. Sensor accuracy vs training instances.

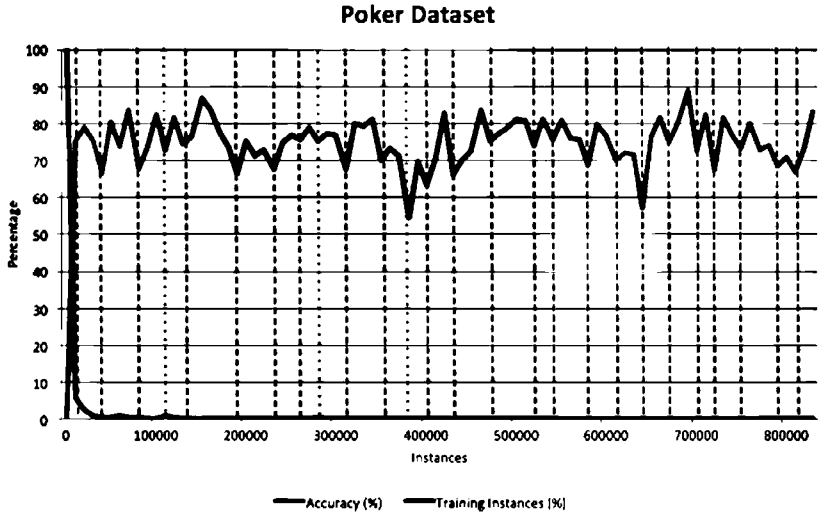

Fig. 8. Poker accuracy vs training instances.

5.4.3. Experiment 3

The goal of this experiment is to prove that MM-PRec provides better precision results than an Active Classifier using the same percentage of instances. In order to do so, apart from the MM-PRec method, an Active Classifier method presented in Žliobaitė et al. (2011) in the assessment of this experiment. This Active Classifier is parametrized to use the same percentage of instances than MM-PRec.

When using the SEA and Airlines datasets, MM-PRec does not provide a great added value when comparing it with the Active Classifier as long as both methods provide similar precision values, as it is shown in Table 5. In the case of the Airlines dataset, MM-PRec

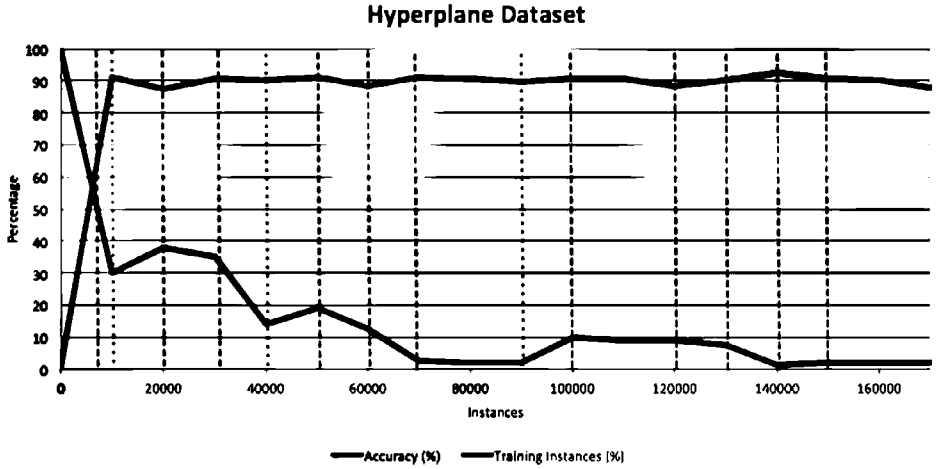

Fig. 9. Hyperplane accuracy vs training instances.

Table 5
Active classifier and MM-PRec comparison.

Dataset	Inst. used (%)	Active classifier			MM-PRec		
		Acc.	Kappa	Time(s)	Acc.	Kappa	Time(s)
SEA	40	83.54 ± 2.16	65.21 ± 4.45	334.78	**84.16 ± 1.14**	**66.49 ± 2.4**	2675.81
Airlines	93	**64.91 ± 5.39**	**21.96 ± 6.52**	384.98	64.85 ± 5.36	20.92 ± 9.01	2005.94
Elec2	88	79.47 ± 7.07	56.81 ± 14.49	0.78	**85.2 ± 3.51**	**69.05 ± 7.23**	264.88
Poker	4	59.4 ± 13.9	14.9 ± 13.31	8.97	**74.44 ± 6.61**	**40.64 ± 16.65**	12194.9
Sensor	85	52.12 ± 20.3	50.96 ± 20.79	271.13	**85.72 ± 13.76**	**85.31 ± 14.21**	15697.42
Hyperplane	60	84.47 ± 6.89	68.9 ± 13.78	3.67	**90.09 ± 1.35**	**80.17 ± 2.69**	5682

provides slightly fewer precision values because of the use of previously seen concepts that do not exactly fit the actual concept. These behaviors reinforces the idea that not existing a big number of gradual drifts the concepts can be represented effectively by the Active Classifier with fewer training instances selected randomly.

Regarding the rest of datasets, the fact is that in all cases MM-PRec provides better precision measures than the Active Classifier. In most cases this improvement is due to the existence of gradual concept drifts. In those cases, the Active Classifier cannot deal with in an effective manner by selecting instances randomly.

To sum up, we can conclude here that MM-PRec provides generally better precision results than an Active Classifier, mostly in those cases where gradual drifts appear.

5.5. Statistical analysis

To validate the results obtained when executing the aforementioned experiments, a statistical analysis has been developed.

In order to do so, the results presented in the previous sections have been used.

5.5.1. Hypothesis

The main goal of this statistical analysis is to prove the following hypothesis:

- $H1$: There is a low rate variation when comparing the precision values of MM-PRec and those provided by the different methods evaluated in experiment 1 (HT-ADWIN, AUE, RCD and MRec).
- $H2$: There is a high rate variation when comparing the precision values of MM-PRec and those of the Active Classifier in particular.
- $H3$: The training records needed by MM-PRec are significant fewer than the ones needed by MRec.

5.5.2. Statistical tests

In order to prove the aforementioned hypothesis, the following tests have been used:

1. The Friedman test (Friedman, 1937) is used based on Demšar (2006) regarding the most suitable tool to compare several methods over multiple datasets, and a post-hoc analysis is developed. Friedman test is safer than parametric tests since it does not assume normal distributions or homogeneity of variance. This is a non-parametric statistical test similar to the parametric repeated measures ANOVA, it is used to detect differences in treatments across multiple test attempts. The procedure involves ranking each row (or block) together, then considering the values of ranks by columns.
2. The Wilcoxon signed ranks test (Wilcoxon, 1945) according to (Demšar, 2006) is a test to be used when dealing with two methods comparison over multiple datasets. The Wilcoxon test is a non-parametric test which ranks the differences in performances of two classifiers for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences. This test is characterized by its sensitivity. It assumes commensurability of differences, but only qualitatively: greater differences still count more, which is probably desired, but the absolute magnitudes are ignored. From the statistical point of view, the test is safer since it does not assume normal distributions. Also, the exceptionally good or bad performances on a few data sets have low effect on the test.

Friedman test has been used in this work to validate hypothesis $H1$. Wilcoxon test has been used to validate hypothesis $H2$ and $H3$.

5.5.3. Results

The results obtained by the execution of the different statistical tests with a significance level of 0.05 are as follows:

1. When executing Friedman test for the precision values related to hypothesis $H1$, a *p-value* of 0.079 is obtained. Therefore, as the *p-value* is greater than the significance level, the results were not significant and no post-hoc analysis is needed. As a consequence, the hypothesis $H1$ can be validated.

2. When executing the Wilcoxon signed-ranks test to evaluate the difference between the precision values of MM-PRec and the Active Classifier (hypothesis $H2$) the *p-value* returned is 0.49×10^{-3}, which demonstrates that the precision values of both methods are significantly different. In fact, MM-PRec provides better precision results than the Active Classifier when using the same percentage of instances. Consequently, the hypothesis H2 is statistically validated.
3. When executing the Wilcoxon signed-ranks test to evaluate the training instances values of both MM-PRec and MRec (hypothesis $H3$) the *p-value* obtained is 0.48×10^{-3}, which demonstrates that the training instances of both methods are significantly different. Consequently, the hypothesis H3 is statistically validated.

6. Conclusions and future lines of research

In this paper the MM-PRec system, an extension of MRec, has been described as a mechanism to deal with concept drift in recurring situations.

The main contributions of MM-PRec are:

1. The implementation of a new similarity concept function using fuzzy logic techniques, which helps in the assessment of similarity between concepts in an improved way.
2. The implementation of HMM as a meta-model representing the underlying concept drifts detected throughout the lifetime of the base learner, by means of a sequence classification mechanism as it is the case of Hidden Markov Models.
3. The development of MM-PRec as a wrapper mechanism, allowing it to be used in an easy way with different base learners and drift detector methods. Furthermore, this wrapper mechanism allows the behavior of the meta-model and the similarity concept function to be parametrized depending on the needs of each dataset or real-world environment.

MM-PRec has been tested on different synthetic and real datasets, and comparisons have been made with other similar context-aware algorithms. The main conclusions obtained from those experiments are that:

- MM-PRec needs less training instances as long as it reuses previously seen models. This aspect has been validated by the experiment 2.
- The fuzzy similarity function helps to find the most appropriate model without loosing precision. This aspect has been validated by experiment 1.
- MM-PRec does not decrease the precision values of the model obtained. This aspect has been validated by experiments 1 and 3.

When comparing MM-PRec with other similar methods, like MRec, RCD or AUE, we can state that in most cases MM-PRec is able to behave in a smoother way, maintaining great precision values, specially in cases where gradual drifts are common. Furthermore, MM-PRec is able to reduce the training instances during the learning process, improving also the repository management in the case of MRec. However, it lacks on efficiency when dealing with abrupt datasets, as long as the meta-model is not useful in most of those cases.

In the particular case of the MRec method presented in Gomes et al. (2010) that relies on a single classifier to represent context-concept relations and on a crisp similarity function to calculate the equivalence between models, it has been proved that MM-PRec meta-model concept-context representation is able to better deal with concept recurrence. Furthermore, when gradual drifts appear, MM-PRec is able to better adapt to them maintaining appropriate precision values. Furthermore, in the specific case of the classification models repository, MM-PRec improves also its management comparing with the work done by MRec thanks to the fuzzy similarity function.

Regarding the behavior of MM-PRec when comparing it with RCD method, we can state that MM-PRec improves in most cases the precision values while reducing the training instances needed. Moreover, it is important to note that just in two cases (Airlines and Poker datasets) the accuracy is better using RCD. This is due to the presence of abrupt drifts. As long as RCD behaves in a better way when abrupt drifts appear, and in contrast the strong point of MM-PRec is dealing with gradual drifts, RCD is able to better fit its behavior with this specific dataset.

When comparing MM-PRec with MRec and RCD, we can state that the former behaves better than the other when dealing with gradual drifts. This is because the meta-model is able to get the most of the context information associated to drifts, and in gradual cases the context information is greater. However, thanks to the fuzzy similarity function, in cases where the meta-model is not used (abrupt drift mainly), MM-PRec is able to provide similar precision values to those provided by MRec and RCD.

The main drawback of MM-PRec though, is the extra computational resources required to train the meta-model. However, this is an obvious fact and the main goal of the paper was to show the feasibility of the solution in terms of reducing the number of records trained without loosing precision, what has been validated by the experimental work presented. In order to deal with the performance issues we are currently working on implementations of the approach in a distributed environment.

Finally, some future lines of research from the work presented in this paper are: (i) analysis of mechanisms to improve performance of the method, in particular the behavior in a distributed environment (ii) analysis of a loss function to improve the accuracy of HMM model.

References

Bicego, M., Murino, V., & Figueiredo, M. A. T. (2004). Similarity-based classification of sequences using hidden markov models. *Pattern Recognition, 37*(12), 2281–2291.

Bifet, A., & Frank, E. (2010). Sentiment knowledge discovery in twitter streaming data. In *Proceedings of the 13th international conference on discovery science. DS'10* (pp. 1–15). Berlin, Heidelberg: Springer-Verlag.

Bifet, A., & Gavaldà, R. (2009). Adaptive learning from evolving data streams. In N. Adams, C. Robardet, A. Siebes, & J.-F. Boulicaut (Eds.), *Advances in Intelligent Data Analysis VIII. Lecture Notes in Computer Science: vol. 5772* (pp. 249–260). Springer.

Blasiak, S., & Rangwala, H. (2011). A hidden markov model variant for sequence classification. In *Proceedings of the twenty-second international joint conference on artificial intelligence - Volume Volume Two. IJCAI'11* (pp. 1192–1197). AAAI Press.

Brzezinski, D., & Stefanowski, J. (2011). Accuracy Updated Ensemble for data streams with concept drift. In E. Corchado, M. Kurzynski, & M. Wozniak (Eds.), *Proceedings of the hybrid artificial intelligent systems. Lecture Notes in Computer Science: vol. 6679* (pp. 155–163). Springer.

Brzezinski, D., & Stefanowski, J. (2013). Reacting to different types of concept drift: the accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems, 25*(1), 81–94.

Cingolani, P., & Alcala-Fernandez, J. (2012). jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation. In *Proceedings of the Ieee international conference on fuzzy systems (fuzz-ieee)* (pp. 1–8).

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement, 20*(1), 37.

Cox, E. (1992). Fuzzy fundamentals. *Spectrum, IEEE, 29*(10), 58–61.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research, 7*, 1–30.

Dietterich, T. G. (2002). Machine learning for sequential data: a review. In T. Caelli, A. Amin, R. Duin, D. Ridder, & M. Kamel (Eds.), *Structural, Syntactic, and Statistical Pattern Recognition. Lecture Notes in Computer Science: vol. 2396* (pp. 15–30). Springer Berlin Heidelberg.

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD international conference on knowledge discovery and data mining. KDD '00* (pp. 71–80). New York, NY, USA: ACM.

Eickeler, S., Kosmala, A., & Rigoll, G. (1998). Hidden markov model based continuous online gesture recognition. In *International conference on pattern recognition (icpr): vol. 2* (pp. 1206–1208).

Elwell, R., & Polikar, R. (2011). Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks, 22*(10), 1517–1531.

Forkan, A. R. M., Khalil, I., Tari, Z., Foufou, S., & Bouras, A. (2015). A context-aware approach for long-term behavioural change detection and abnormality prediction in ambient assisted living. *Pattern Recognition, 48*(3), 628–641.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*(200), 675–701.

Gaber, M., Zaslavsky, A., & Krishnaswamy, S. (2007). A survey of classification methods in data streams. In A. K. Elmagarmid, & C. C. Aggarwal (Eds.), *Data Streams - Models and Algorithms. Advances in Database Systems: vol. 31* (pp. 39–59). Springer.Chapter 3

Galili, T. (2010). *Post-hoc analysis for Friedman test*.Code available in http://www.r-statistics.com/2010/02/post-hoc-analysis-for-friedmans-test-r-code.

Gama, J., & Kosina, P. (2009). Tracking Recurring Concepts with Meta-learners. In *Progress in artificial intelligence. Lecture Notes in Computer Science: vol. 5816* (pp. 423–434).

Gama, J., Medas, P., Castillo, G., Rodrigues, P. P., & Labidi, S. (2004). Learning with drift detection. In A. L. C. Bazzan, & S. Labidi (Eds.), *Proceedings of SBIA 2004 advances in artificial intelligence. Lecture Notes in Computer Science: vol. 3171* (pp. 286–295). Springer.

Gama, J. a., & Kosina, P. (2014). Recurrent concepts in data streams classification. *Knowledge and Information Systems, 40*(3), 489–507.

Gama, J. a., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM. Computing Surveys, 46*(4), 44:1–44:37.

Ghoshal, A., Ircing, P., & Khudanpur, S. (2005). Hidden markov models for automatic annotation and content-based retrieval of images and video. In *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval. SIGIR '05* (pp. 544–551). New York, NY, USA: ACM.

Gillies, M. (2010). http://www.r-statistics.com/2010/02/post-hoc-analysis-for-friedmans-test-r-code. Accessed 28.10.15.

Gomes, J. B., Menasalvas, E., & Sousa, P. A. C. (2011). Learning recurring concepts from data streams with a context-aware ensemble. In *Proceedings of the 2011 ACM symposium on applied computing. SAC '11* (pp. 994–999). New York, NY, USA: ACM.

Gomes, J. B., Sousa, P. A. C., & Menasalvas, E. (2010). Tracking recurrent concepts using context. In *Rough Sets and Current Trends in Computing. Lecture Notes in Computer Science: vol. 6086* (pp. 168–177). Springer.

Gonçalves Jr, P. M., & Barros, R. S. (2013). Rcd: a recurring concept drift framework. *Pattern Recognition Letters*.

Haque, A., Parker, B., Khan, L., & Thuraisingham, B. (2014). Evolving big data stream classification with mapreduce. In *Proceedings of the 2014 IEEE international conference on cloud computing. CLOUD '14* (pp. 570–577). Washington, DC, USA: IEEE Computer Society.

Harries, M. (1999). Splice-2 comparative evaluation: electricity pricing. *Technical report*. The University of South Wales.

Harries, M., Sammut, C., & Horn, K. (1998). Extracting hidden context. *Machine Learning, 32*(2), 101–126.

Hewahi, N. M., & Elbouhissi, I. M. (2015). Concepts seeds gathering and dataset updating algorithm for handling concept drift. *International Journal of Decision Support System Technology, 7*(2), 29–57.

Hollander, M., & Wolfe, D. A. (1999). *Nonparametric Statistical Methods*.

Holmes, G., Kirkby, R., & Pfahringer, B. (2007). *MOA: Massive Online Analysis, 2007*. http://sourceforge.net/projects/moa-datastream/.

Hosseini, M. J., Ahmadi, Z., & Beigy, H. (2012). New management operations on classifiers pool to track recurring concepts. In *Data Warehousing and Knowledge Discovery* (pp. 327–339). Springer.

Hu, J., Brown, M., & Turin, W. (1996). Hmm based online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18*(10), 1039–1045.

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the seventh acm sigkdd international conference on knowledge discovery and data mining* (pp. 97–106).

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2010). Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowledge and Information Systems, 22*(3), 371–391.

Kosina, P., & Gama, J. a. (2015). Very fast decision rules for classification in data streams. *Data Mining and Knowledge Discovery, 29*(1), 168–202.

Kosinski, W., Prokopowicz, P., & Slezak, D. (2005). Calculus with fuzzy numbers. In *Proceedings of the second international conference on intelligent media technology for communicative intelligence. IMTCI'04* (pp. 21–28). Berlin, Heidelberg: Springer-Verlag.

Lane, T., & Brodley, C. (1999). Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security, 2*(3), 295–331.

Mena-Torres, D., & Aguilar-Ruiz, J. S. (2014). A similarity-based approach for data stream classification. *Expert Systems With Applications, 41*(9), 4224–4234.

Mendel, J. (1995). Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE, 83*(3), 345–377.

Muhlbaier, M., Topalis, A., & Polikar, R. (2009). Learn++. NC: combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. *IEEE Transactions on Neural Networks, 20*(1).

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 257–286.

Ramamurthy, S., & Bhatnagar, R. (2007). Tracking recurrent concept drift in streaming data using ensemble classifiers. In *Proceedings of the sixth international conference on machine learning and applications* (pp. 404–409). IEEE Computer Society.

Street, W. N., & Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 377–382). New York, NY, USA: ACM.

R Development Core Team (2010). R: a language and environment for statistical computing. In *R foundation for statistical computing, vienna, austria*.ISBN 3-900051-07-0.

Tsymbal, A. (2004). *The problem of concept drift: definitions and related work*. Computer Science Department, Trinity College Dublin.

Turney, P. (1993). Exploiting context when learning to classify. In P. Brazdil (Ed.), *Machine Learning: ECML-93. Lecture Notes in Computer Science: vol. 667* (pp. 402–407). Springer.

Žliobaitė, I. (2010). Learning under concept drift: an overview. *Technical report*. Faculty of Mathematics and Informatics, Vilnius University: Vilnius, Lithuania.

Žliobaitė, I., Bifet, A., Gaber, M. M., Gabrys, B., Gama, J., Minku, L. L., et al. (2012). Next challenges for adaptive learning systems. *SIGKDD Explorations, 14*(1), 48–55.

Žliobaitė, I., Bifet, A., Pfahringer, B., & Holmes, G. (2011). Active learning with evolving streaming data. In *Proceedings of the 2011 european conference on machine learning and knowledge discovery in databases - Volume Part III. Lecture Notes in Computer Science: vol. 6913* (pp. 597–612). Berlin, Heidelberg: Springer.

Žliobaitė, I., Bifet, A., Read, J., Pfahringer, B., & Holmes, G. (2015). Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning, 98*(3), 455–482.

Widmer, G. (1997). Tracking context changes through meta-learning. *Machine Learning, 27*(3), 259–286.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning, 23*(1), 69–101.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin, 1*(6), 80–83.

Xing, Z., Pei, J., & Keogh, E. (2010). A brief survey on sequence classification. *SIGKDD Explorations Newsletter, 12*(1), 40–48.

Yang, Y., Wu, X., & Zhu, X. (2005). Combining proactive and reactive predictions for data streams. In R. Grossman, R. Bayardo, & K. P. Bennett (Eds.), *Proceedings of the eleventh acm sigkdd international conference on knowledge discovery in data mining, new york, ny, usa* (pp. 710–715).

Yang, Y., Wu, X., & Zhu, X. (2006). Mining in anticipation for concept change: proactive-reactive prediction in data streams. *Data mining and knowledge discovery, 13*(3), 261–289.

Zadeh, L. (1965). Fuzzy sets. *Information and Control, 8*(3), 338–353.

Zhu, X. (2010). *Stream Data Mining Repository*. http://www.cse.fau.edu/~xqzhu/stream.html.

Zliobaite, I., Bifet, A., Gaber, M., Gabrys, B., Gama, J., Minku, L., et al. (2012). Next challenges for adaptive learning systems. *SIGKDD Explorations Newsletter, 14*(1), 48–55.